Université de Montréal

From Word Embeddings to Large Vocabulary Neural Machine Translation

par
Sébastien Jean

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

Avril, 2015

RÉSUMÉ

Dans ce mémoire, nous examinons certaines propriétés des représentations distribuées de mots et nous proposons une technique pour élargir le vocabulaire des systèmes de traduction automatique neurale. En premier lieu, nous considérons un problème de résolution d'analogies bien connu et examinons l'effet de poids adaptés à la position, le choix de la fonction de combinaison et l'impact de l'apprentissage supervisé. Nous enchaînons en montrant que des représentations distribuées simples basées sur la traduction peuvent atteindre ou dépasser l'état de l'art sur le test de détection de synonymes TOEFL et sur le récent étalon-or SimLex-999. Finalament, motivé par d'impressionnants résultats obtenus avec des représentations distribuées issues de systèmes de traduction neurale à petit vocabulaire (30 000 mots), nous présentons une approche compatible à l'utilisation de cartes graphiques pour augmenter la taille du vocabulaire par plus d'un ordre de magnitude. Bien qu'originalement développée seulement pour obtenir les représentations distribuées, nous montrons que cette technique fonctionne plutôt bien sur des tâches de traduction, en particulier de l'anglais vers le français (WMT'14).

Mots clés: Analogies, word2vec, TOEFL, SimLex-999, accélération, apprentissage profond, GPU, WMT.

ABSTRACT

In this thesis, we examine some properties of word embeddings and propose a technique to handle large vocabularies in neural machine translation. We first look at a well-known analogy task and examine the effect of position-dependent weights, the choice of combination function and the impact of supervised learning. We then show that simple embeddings learnt with translational contexts can match or surpass the state of the art on the TOEFL synonym detection task and on the recently introduced SimLex-999 word similarity gold standard. Finally, motivated by impressive results obtained by small-vocabulary (30,000 words) neural machine translation embeddings on some word similarity tasks, we present a GPU-friendly approach to increase the vocabulary size by more than an order of magnitude. Despite originally being developed for obtaining the embeddings only, we show that this technique actually works quite well on actual translation tasks, especially for English to French (WMT'14).

Keywords: Analogies, word2vec, TOEFL, SimLex-999, speed-up, deep learning, GPU-friendly, WMT

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

BilBOWA	Bilingual Bag-of-Words without Alignments
BLEU	Bilingual Evaluation Understudy
BOW	Bag of Words
CBOW	Continuous Bag of Words
GloVe	Global Vectors
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
LM	Language Model
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
MSR	Microsoft Research
MT	Machine Translation
NLP	Natural Language Processing
NMT	Neural Machine Translation
PDW	Position-Dependent Weights
RNN	Recurrent Neural Network
SMT	Statistical Machine Translation
TOEFL	Test of English as a Foreign Language
t-SNE	t-Distributed Stochastic Neighbour Embedding
UNK	Unknown Word
VSM	Vector Space Model
WMT	Workshop on Statistical Machine Translation

NOTATION

$\cdot$	Dot Product
$\odot$	Element-wise Product
∇	Gradient
σ	Logistic Sigmoid Function
$\|\|\cdot\|\|^2$	Norm Squared
$p(x\|y)$	Probability of x given y
$\vec{v}$	Vector

ACKNOWLEDGMENTS

I would foremost like to thank my advisor Roland Memisevic, who has guided me throughout the course of my studies and with whom I had many enlightening conversations. I am also very grateful to Kyunghyun Cho, who has been extremely kind, supportive and resourceful. I would also like to thank Yoshua Bengio for spearheading at MILA a large effort in advancing the state of machine learning. I am furthermore thankful to my thesis committee members, in particular to Philippe Langlais for his thoughtful comments.

My experience at MILA would not have been as enriching without every student I had to opportunity to interact with. I would especially like to thank Dzmitry Bahdanau and Felix Hill for their inspiring work. Thanks also to Orhan Firat, with whom it has been a pleasure to collaborate.

I would also like to thank NSERC for funding my studies and the MILA lab for lending me a GPU at home, which has helped catalyze my research efforts.

I would finally like to thank my family for their unwavering support.

CHAPTER 1

INTRODUCTION

Good representations are often crucial to the success of machine learning [11]. For example, in computer vision, while classification from raw pixels is only very mildly successful, convolutional nets [62] convert the original image into much more informative and discriminative features.

In language, while we may represent words as atomic units, this is somewhat problematic as no information is shared between similar or related entities. However, if for example we know that *cats* and *dogs* have much in common, leveraging that information could help generalize better about unknown facts. To allow for such knowledge transfer, words, or more complex linguistic units such as sentences and documents, may be *embedded* in a common vector space where closely related entities will be nearby each other [10, 50].

Such distributed representations of words have proven very useful in many natural language processing (NLP) tasks. In addition to language modeling [10, 73], part-of-speech tagging, chunking, semantic role labelling, named entity recognition, parsing, sentiment analysis and machine translation have been successfully tackled with neural networks built upon word embeddings [23, 96, 97, 99, 100]. Given the importance of these word representations, we try to understand their properties better in much of this thesis. More precisely, chapter 5 describes various experiments related to word analogy tasks, while we examine simple multilingual embeddings in chapter 6.

When using such neural networks with large output spaces, as is common in NLP, the last *softmax* layer is often a significant bottleneck slowing training and inference significantly. As such, sidestepping this issue presents obvious advantages. For this reason, another substantial part of this thesis addresses this problem under the recent framework of neural machine translation (NMT) [21, 54, 99]. This work, which was accepted as an article at ACL 2015, is presented in chapter 8.

Prior to these three core chapters, we situate our work by discussing language models

and some basics of machine learning (chapter 2), alternatives to the softmax (chapter 3) and word embeddings (chapter 4).

CHAPTER 2

LANGUAGE MODELING

The quantity of sentences that will ever be written or uttered is gigantic, yet some are more likely than others. While sentences such as "How are you?" are common, especially in speech, others such as "The blue sun plays music." are not proper English even if they may be grammatically correct. Assigning probabilities to sentences is known as *language modeling* and has proven helpful in many fields such as speech recognition or machine translation [19].

2.1 N-gram language models

Language models are generally learnt from a training corpus containing millions to billions of words. A simplistic strategy would be to work at the sentence level and assign some probability mass to all seen sentences, and none to all others. However, for such an approach to be even moderately successful, the amount of data needed would be gigantic.

Instead of using complete sentences, it is possible to rely on the chain rule of probability, which tells us that the probability of any sequence (w_1, w_2, ..., w_m) can be decomposed as

$$p(w_1,...,w_m) = p(w_1)p(w_2|w_1)p(w_3|w_1,w_2)...p(w_m|w_1,...,w_{m-1}) \tag{2.1}$$

Due to constraints such as data sparsity and memory limitations, *n-gram* models approximate such probabilities by considering histories of only $n-1$ tokens, with n generally between 2 and 5. Using (w_i^{i-n+1}) as a shorthand for the n-gram $(w_{i-n+1},...,w_i)$, the previous equation would become

$$p(w_1^m) \approx \prod_{i=1}^{m} p(w_i|w_{i-n+1}^{i-1}) \tag{2.2}$$

In practice, the sentence is padded by $n-1$ <bos> tokens on the left and one <eos> token on the right. This allows the model to explicitly take into account the position of the first $n-1$ tokens, as well as ensuring that the probabilities of all sentences sum to 1 [19].

For a specified order n, a simple n-gram model would then assign probabilities according to

$$p_{ML}(w_k|w_{k-n+1}^{k-1}) = \frac{c(w_{k-n+1}^{k})}{c(w_{k-n+1}^{k-1})} \tag{2.3}$$

where $c(.)$ is the number of occurrences of a given n-gram. Such an approach maximizes the likelihood of the training data for a given n.

2.1.1 Evaluation

Would such a model be any good? Language models can be evaluated either extrinsically or intrinsically. In the former case, which is arguably the most important, the quality of a language model is measured by the improvements (or lack thereof) on some meaningful tasks such as speech recognition, spelling correction or machine translation. In the second case, the model is evaluated on its own, using a metric such a perplexity, which is defined as

$$PP(w_1^N) = (p_{LM}(w_1^N))^{1/N} \tag{2.4}$$

To get an unbiased estimate of the quality of a model, it is essential that perplexity be evaluated on a test set that is distinct from the training data used to build the model. In particular, we want models that generalize well to new previously unseen data.

N-gram models trained with maximum likelihood estimation face a serious problem when encountering an n-gram that was not in the training set. For example, in speech recognition, some correct transcriptions will be wrongly discarded if the language model assigns them zero probability. Moreover, the perplexity of such a model will be infinite.

2.1.2 Smoothing

To address this issue, maximum-likelihood probabilities are *smoothed* by transferring some probability mass from actual events to rare or unseen n-grams. A common framework for many smoothing techniques is interpolation, where n-grams models of different orders are weighted to give the final model probability. Such models can be defined recursively as

$$p_{LM}(w_i|w_{i-n+1}^{i-1}) = \alpha(w_i|w_{i-n+1}^{i-1}) + \gamma(w_{i-n+1}^{i-1})p_{LM}(w_i|w_{i-n+2}^{i-1}) \tag{2.5}$$

In general, the γ's are conditioned on the context, which allows the model to trust long or short histories more appropriately. An alternative to interpolated models are backoff models, where the lower-order models would be used only for unseen n-grams.

A good technique still in wide use today is "modified Kneser-Ney smoothing" [19, 58]. In this case, the α's are defined as

$$\alpha(w_i|w_{i-n+1}^{i-1}) = \frac{c_{KN}(w_{i-n+1}^{i}) - D(c_{KN}(w_{i-n+1}^{i}))}{\sum_w c_{KN}(w_{i-n+1}^{i-1}w)} \tag{2.6}$$

where $c_{KN}(w_a^b)$ is the actual count for the higher-order n-gram model, and is otherwise the number of distinct words that w_a^b follows in the training data. For example, this allows the model to give low unigram continuation probability to words such as "Francisco" or "Canadiens", which will respectively often occur after "San" and "Montreal / the", but very rarely after other words. Allowing D to depend on the count of the n-gram it discounts was found to be beneficial by Chen and Goodman [19].

2.2 Monolingual neural language models

In addition to counting n-grams, it is also possible to use neural networks to model language. We first present multiple architectures used to build such models and discuss how to train them. We then follow with multiple approaches for speeding up these language models, either during training or at test time.

2.2.1 Feedforward neural language models

In a standard feed-forward neural language model [9], all $n-1$ context input words are first projected into a common d-dimensional vector space with an embedding matrix E.

x_0, the concatenation of these $n-1$ vectors, is then transformed into another vector x_k through k hidden layers . In the most common case, this transformation is defined by a sequence of functions

$$x_{i+1} = f_i(W_i x_i + b_i) \tag{2.7}$$

where W_i is a weight matrix and b_i a bias vector. f_i is a non-linearity such as the hyperbolic tangent $f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$ or the rectifier function $f(x) = max(0,x)$.

With a final weight matrix W_{k+1} and bias b_{k+1}, the last hidden state x_k is transformed into a vector $\mathscr{E}$ of dimension $|V|$, where the i^{th} element of $\mathscr{E}$ can be viewed as a score for the i^{th} word (the higher the better). These scores can be converted to probabilities with the softmax function

$$p(i) = \frac{\exp(\mathscr{E}^{(i)})}{\sum_{j=1}^{|V|}\exp(\mathscr{E}^{(j)})} = \frac{\exp(\mathscr{E}^{(i)})}{Z} \tag{2.8}$$

where Z, the sum of all exponentiated scores, is known as the partition function.

Given N words to predict, the cost function generally associated with such a neural language model is

$$C(E, W_1, ..., W_{k+1}, b_1, ..., b_{k+1}) = -\frac{1}{N}\sum_{i=1}^{N}\log(p(w_i|w_{i-n+1}^{i-1})) \tag{2.9}$$

This cost [1], known as the cross-entropy loss, is simply the negative logarithm of the perplexity (see equation 2.4). In some cases, it may be advisable to add a regularization term R to this cost. Such a term can prevent the weights of the network to grow too big and may lead to a better generalization error on a new unseen test set.

1. For simplicity, the notation of equation 2.9 does not explicitly consider the <bos> tokens added at the beginning of each sentence.

2.2.2 Recurrent neural network language models

Another alternative for language modeling is the use of recurrent neural networks [72] which, contrarily to feedforward nets, can in principle model arbitrarily long sequences. As with feedforward architectures, words are projected into a common low-dimensional vector space. However, rather than using a fixed number of previous words to predict the next one, a recurrent neural network will define the hidden state h_{t+1} as a function of the previous hidden state h_t and well as the current token w_{t+1}. For a simple recurrent neural network, hidden states are commonly defined as

$$h_{t+1} = \tanh(Uh_t + Ew_{t+1} + b), \quad t \geq 1 \tag{2.10}$$

where h_0 could be, for example, an all-zero vector. Through the use of another weight matrix and bias vector, each hidden state is transformed into a score vector, which is converted to probabilities as previously.

One possible issue which such simple recurrent neural networks is that the hidden state is overwritten at every time step, which may make it harder to remember useful long-term information. A common solution to this issue if to use gated units such as the long short term memory (LSTM) [42, 52] or the more recent gated recurrent unit (GRU) [21], where it is possible to better control how the hidden state is modified through time.

More precisely, an LSTM block is [2]

$$\begin{aligned}
i_{t+1} &= \sigma(U^{(i)}h_t + W^{(i)}Ew_{t+1} + b^{(i)}) \\
f_{t+1} &= \sigma(U^{(f)}h_t + W^{(f)}Ew_{t+1} + b^{(f)}) \\
o_{t+1} &= \sigma(U^{(o)}h_t + W^{(o)}Ew_{t+1} + b^{(o)}) \\
z_{t+1} &= \tanh(U^{(z)}h_t + W^{(z)}Ew_{t+1} + b^{(z)}) \\
c_{t+1} &= f_{t+1} \odot c_t + i_{t+1} \odot z_{t+1} \\
h_{t+1} &= o_{t+1} \odot \tanh(c_{t+1})
\end{aligned} \tag{2.11}$$

2. There are multiple variants of LSTM. We present the one used in [42]

where σ is the logistic sigmoid function and $\odot$ denotes element-wise multiplication.

The central component of the LSTM is the cell c, which stores the information over time. It is updated as an additive combination of its previous self and of a new proposal z, with their respective contribution controlled by the forget gate f and the input gate i. As such, an LSTM can keep some information unchanged (or barely modified), ignore some inputs or erase the history if such actions are helpful. There is also an output gate o, which controls how much of the cell is exposed at the next time step.

2.3 Bilingual neural language models

In addition to traditional language modeling, neural networks may also estimate the probability of a sentence given some external context. In particular, by conditioning their output on a source sentence, neural machine translation (NMT) systems are able to translate text into many target languages. While most work has focused on improving statistical machine translation (SMT) systems [21, 26, 54], rapid progress has enabled NMT systems to generate translations that are competitive with the best phrase-based or syntax-based approaches [2, 53, 71, 99].

2.3.1 Feedforward bilingual language models

To predict the probability of a word in a translation setting, Devlin et al. [26] proposed a bilingual extension of feedforward language models. Rather than simply using the few previously generated words as context, they also consider the source word most affiliated to the token to predict as well as some of its neighbours. Even though this approach is quite simple, very impressive improvements were observed when integrating such models into statistical translation systems.

2.3.2 Encoder-decoder approaches

To perform end-to-end translation with neural networks, the current framework is based on encoder-decoder approaches [21, 54, 99]. The source sentence is first encoded into one or multiple vectors, either using convolutional [65] or recurrent neural networks.

Given this representation(s) of the source sentence, another recurrent network, called the decoder, is tasked to generate each word one by one. To do so, it condition its output on the previously generated word, the current hidden state as well as on a representation of the source sentence. While using a global representation may work well for very large networks, especially if the sentence was read in reverse order [99], a clever approach that lets the network choose where to look within the source sentence has been proposed by Bahdanau et al. [2] With this alignment-based encoder-decoder, translations of similar quality may be obtained from much smaller networks.

2.3.3 Evaluating machine translation

Although the ultimate gold standard to assess the quality of machine-generated output is human evaluation, doing so is expensive and not very practical when designing new systems. As such, having a fast automated proxy is very desirable. While many metrics may be used to evaluate machine translation systems, the most common one is BLEU [88]. The guiding principle behind the design of BLEU is that a good translation should look similar to one that was produced by a human. BLEU is a precision-based metric that counts, for $n = 1$ to 4, the number of n-grams in the generated translation that are also found in the reference [3]. Instead of using the usual precision (number of matches divided by total number of n-grams), BLEU modifies it so that n-gram that are repeated too frequently are penalized. For example, if 'the' was present 7 times in a translation but only twice in the reference, the modified precision for that word would be $\frac{2}{7}$ instead of 1 [88]. As precision-based metrics could be tricked by producing very short output, BLEU introduces a brevity penalty, which lowers the scores for translations that are shorter than the reference. For a translation and reference of lengths t and r respectively, this brevity penalty is defined as

$$BP = min(\exp(1 - r/t), 1) \tag{2.12}$$

3. BLEU is designed to work with multiple references, but we only consider one to simplify presentation.

Finally, given modified precisions p_1, p_2, p_3 and p_4, BLEU computes the score of a translation as

$$BLEU = BP \cdot \exp\left(\sum_{i=1}^{4} \frac{1}{4} \log p_i\right) \tag{2.13}$$

2.4 Training neural language models

For a neural network to be useful, its parameters must be adjusted to the task at hand. Fortunately, for both feedforward and recurrent neural language models, as well as for the vast majority of other neural networks, the cost C is differentiable with respect to the parameters θ, which generally include all weight matrices and bias vectors of the model. As such, a simple learning strategy is to use gradient descent

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} C \tag{2.14}$$

where η is a learning rate that must be adjusted and possibly changed over time. If it is too small, the cost will diminish slowly, whereas if it is too big, the optimization may fail to converge to a low-cost solution.

For large datasets such as those used in language modeling, using gradient descent as is would lead to slow progress since the parameters would be updated only once every pass over the data, which is known as an epoch. As the cost is defined as a sum (or an average) over all the training examples, a common solution is to split the training data in many mini-batches consisting of a few examples each. For each of these mini-batches, a cost can be computed, which allows the model parameters to be updated. This strategy is known as stochastic (or mini-batch) gradient descent (SGD).

Stochastic gradient descent may often be accelerated by the use of momentum [98], where the parameter updates are given by

$$\begin{aligned} v_{t+1} &= \alpha v_t - \eta \nabla_{\theta_t} C_{t+1} \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned} \tag{2.15}$$

where C_{t+1} is the cost associated to mini-batch $t+1$ and α is chosen between 0 and

1. By using a velocity vector v, gradients that point in the same direction over multiple mini-batches strengthen each other while others will be dampened. Note that by setting $\alpha = 0$, SGD is recovered. Stochastic gradient descent may also be replaced by an adaptive learning algorithm such as Adagrad, Adadelta or Adam [27, 55, 109], where each parameter has its own learning rate.

An issue with all these learning algorithms, especially for recurrent neural networks, is their susceptibility to exploding gradients [8], where the derivatives for some parameters become so large that applying the usual parameter updates would render the network useless. A solution to this problem is clipping gradients [89] that are too big in magnitude to some finite norm, which allows learning to progress normally even after encountering pathological gradients.

2.4.1 Hyper-parameter selection and ensembles

As mentioned in section 2.1.1, the goal of language models, or almost any other trained neural network, is not to perform well on the training set but to generalize to unseen data. Before conducting a final unbiased evaluation on a given test set, many hyper-parameters, which include the learning procedure, the type of regularization and all model architectural details such as the number of layers and their size, must be chosen. To do so, in addition to the training and test set, a validation (or development) set which does not overlap with either of the other two is used to monitor performance. The best model on this set can then be evaluated fairly.

Rather than choosing the best single model, it is possible to use multiple ones simultaneously in an ensemble setting by averaging their probabilities. This approach has proven extremely powerful and was key to set the state of the art on many tasks. For this technique to work best, in addition to being good on their own, the models that are combined should be as diverse as possible [63]. Disadvantages of ensembling include longer training and test times, although there has been recent progress to sidestep some of these issues [51].

CHAPTER 3

ALTERNATIVES TO THE SOFTMAX

The bottleneck of many neural language models is the softmax layer (eq. 2.8) used to predict each word. This step is generally slow because, in order to obtain valid probabilities, all scores must be normalized in a time proportional to the size of the vocabulary, which can easily extend to hundreds of thousands of words. In this chapter, we present multiple alternatives that have been devised to mitigate this issue.

3.1 Class-based and hierarchical softmax

The core idea behind class-based softmax is to put each word of the vocabulary V into one of many classes [40]. Then, given a context h, the probability of word w in class K is decomposed as

$$P(w|h) = P(K|h)P(w|K,h) \tag{3.1}$$

where each probability distribution on the right-hand side of eq. 3.1 can be modeled by a softmax function over a much smaller set of targets. This approach can be extended by constructing a tree over the vocabulary [84], where each word is a leaf. To compute the probability of a token, it suffices to multiply the probabilities of each node from the root towards the corresponding leaf.

The classes or tree can be constructed with a frequency-based approach [74], for example by putting words with similar counts into the same bin or by using a Huffman tree. An arguably better approach is to take semantics into account and put similar words in the same class or close to each other in the tree [13].

During training, class-based softmax can run in time $O(\sqrt{|V|})$ rather than $O(|V|)$, whereas the tree-based hierarchical softmax can improve this to $O(\log(|V|))$. At test time, the same benefits are obtained if the probability of a single word is needed. However, to get the probabilities of all words, there is no gain from these techniques.

3.2 Short-lists

A very simple way to deal with a large output space is to not use one! This is the idea behind short-lists [94], which contain only a fixed subset of the entire vocabulary. In order to assign probabilities to the ignored words, a special <UNK> token may be used, possibly in conjunction with a less computationally expensive n-gram model.

3.3 Importance sampling

In most neural network language models, the gradient of the log-likelihood of each word w can be written in the following form

$$\nabla \mathscr{E}(w,h) - \sum_{w'} P(w'|h) \nabla \mathscr{E}(w',h) \tag{3.2}$$

While computing the positive part $\nabla \mathscr{E}(w,h)$ can be done quickly, doing so for the negative part, $-\sum_{w'} P(w'|h) \nabla \mathscr{E}(w',h)$, is expensive. In importance sampling, this problematic term is replaced by an approximation that can be evaluated quickly [6, 7].

For any random variable $g(X,h)$ and proposal distribution $Q(X|h)$, importance sampling relies on the fact that using M samples from Q,

$$\frac{1}{M} \sum_{x' \sim Q} \frac{P(x'|h)}{Q(x'|h)} g(x',h) \tag{3.3}$$

is an unbiased estimator of $E_P[g(X,h)]$.

Since the negative part can be written as $-E_P[\nabla \mathscr{E}(w',h)]$, given samples $w^{(1)}, ..., w^{(M)}$ from Q, it can be estimated as

$$-\frac{1}{M} \sum_{i=1}^{M} \frac{\exp(\mathscr{E}(w^{(i)},h))}{Z(h) Q(w^{(i)}|h)} \nabla \mathscr{E}(w^{(i)},h) \tag{3.4}$$

One problem with the previous equation is that it still contains the normalization constant. To replace it, we may leverage that $Z(h)$ can also be expressed as an expectation and estimated by 3.3. After substitution, a biased estimator of the negative part is given

by

$$-\frac{1}{\sum_{j=1}^{M} f(w^{(j)})} \sum_{i=1}^{M} f(w^{(i)}) \nabla \mathscr{E}(w^{(i)}, h) \tag{3.5}$$

with $f(w^{(i)}) = \exp(\mathscr{E}(w^{(i)}) - \log(Q(w^{(i)}|h)))$. With this approximation, each gradient descent update now takes time proportional to M rather than $|V|$.

3.4 Noise-contrastive estimation and related techniques

During training, noise-contrastive estimation (NCE) [43, 83] transforms the $|V|$-way classification problem into a binary one, where the new proxy task is to predict whether a word is from the actual data ($D = 1$) or whether it was one of k drawn from some noise distribution P_n ($D = 0$). According to the model P, the probability that a word is real data is given by

$$P_{binary}(D = 1|w,h) = \frac{P(w|h)}{P(w|h) + kP_n(w|h)} \tag{3.6}$$

As we would like the model to distinguish data from noise, for each data point w, we maximize the objective function

$$J = \log(P_{binary}(D = 1|w,h)) + \sum_{i=1}^{k} \log(P_{binary}(D = 0|w^{(i)},h)) \tag{3.7}$$

By itself, this formulation is not sufficient to provide an appreciable speed-up since there is still a need to compute the normalization constant. Instead of modeling $P(w|h)$ as $\exp(\mathscr{E}(w,h))/Z(h)$, noise-contrastive estimation parametrizes it as $\exp(\mathscr{E}(w,h))\exp(c(h))$, where the $c(h)$ may either be learnt or all fixed to 0 with the hope that as training progresses, the model will learn for the actual normalizers $Z(h)$ to be close to 1. At test time, if the probabilities are explicitly normalized, we obtain a true probability distribution, but at a high computational cost. Alternatively, if the model was learned by fixing $c(h) = 0$, using the unnormalized probabilities directly may be good enough for some tasks [105].

A closely related technique is negative sampling, where the objective to maximize is

the same as in 3.7. However, in this case, $P_{binary}(D = 1|w,h)$ is written as

$$P_{binary}(D = 1|w,h) = \frac{\exp(\mathscr{E}(w,h))}{\exp(\mathscr{E}(w,h)) + 1} = \sigma(\mathscr{E}(w,h)) \tag{3.8}$$

While this may not be ideal for language modeling [29], negative sampling may nevertheless be used in situations where obtaining valid probabilities is not crucial, for example to create word embeddings.

3.5 Self-normalization

As mentioned previously, a softmax over many targets is slow because of the need to compute the sum of all scores. As such, if this sum were always 1, the softmax speed issue would be essentially resolved, at least if we only need to know the probability of a few words. Devlin et al. [26] propose complementing the usual loss function with a regularization term that forces the normalizer Z to be close to 1. For a given context h, the training cost would therefore become

$$C = -\frac{1}{N}\sum_{i=1}^{N}\left(\log(p(w_i|h)) - \alpha\log^2(Z(h))\right) \tag{3.9}$$

where α may be chosen on a validation set. If it is too low, the unnormalized probabilities will lead to much worse performance than normalized ones, whereas if it is too high, the scores can become meaningless. While this technique is, in this form, only beneficial at test time, simple modifications make it more efficient during training as well [1].

CHAPTER 4

WORD EMBEDDINGS

One central characteristic of neural language models is that words are represented as vectors instead of atomic units. As such, as learning progresses, similar or related words will often be mapped close to each other. Aside from language modeling, distributed representations may be used, sometimes as part of more complex systems, for many NLP tasks such as part-of-speech tagging, chunking, semantic role labelling, named entity recognition, parsing, sentiment analysis and machine translation [23, 96, 97, 99, 100].

4.1 Count-based word representations

Word representations need not be obtained as by-product of neural networks. In particular, there is a large body of literature on count-based vector space models (VSM) [93, 102], which may be as good as neural (or "neurally-inspired") embeddings [67–69]. These VSM are constructed around the distributional hypothesis [33, 44], which states that "you know a word by the company it keeps". To build such a VSM from a text corpus, each token in the vocabulary is represented as a large sparse vector, where each dimension explicitly corresponds to a specific context. While neighbouring words are often used as contexts, other alternatives, based on either syntax or bilingual data, may be considered [87, 104].

For $|V|$ words and $|C|$ distinct contexts, the simplest way to build a VSM is to build a $|V| \times |C|$ matrix M where each cell $m_{i,j}$ contains the number of times context j was associated to word i. To obtain better representations, it is common practice to modify the raw counts, using a transformation such as PPMI (positive pointwise mutual information) [22, 85], which is defined as

$$PPMI(w,c) = max\left(0, \log\left(\frac{P((w,c))}{P(w)P(c)}\right)\right) \tag{4.1}$$

where the probabilities of each word, context and word-context pairs may be approxi-

mated from data by maximum likelihood estimation (see section 2.1).

4.1.1 Singular value decomposition for dimensionality reduction

To obtain representation that look more like those obtained from neural models, the sparse high-dimensional VSMs may be transformed using singular value decomposition (SVD) [25]. Given a matrix M of size $|V| \times |C|$, SVD decomposes it exactly as

$$M = U\Sigma V^* \tag{4.2}$$

where U and V^* are unitary matrices [1] of size $|V| \times |V|$ and $|C| \times |C|$ respectively. Σ, of size $|V| \times |C|$, is a diagonal matrix of non-negative decreasing singular values. To obtain low d-dimensional word representations, it suffices to use

$$L = U_d \Sigma_d^p \tag{4.3}$$

where U_d and Σ_d are the matrices containing the first d columns of U and d rows of Σ respectively , and where p is a tunable hyper-parameter [17].

4.2 Simple neurally-inspired word embeddings

While deep neural language models may be used to obtain word embeddings, training them on very large corpora may be too slow even with the techniques discussed in section 3. As learning embeddings is a somewhat simpler problem than actual language modeling, recent methods that strip neural networks of most of their complexity have been shown to nevertheless produce high quality word representations.

4.2.1 Skip-gram and CBOW

Skip-gram and CBOW are arguably the most famous "neurally-inspired" embedding algorithms [75, 77]. They are closely related to the log-bilinear language model [79, 80] ,

1. With V^* the conjugate transpose of V and I the identity matrix, $VV^* = I$

which is a simple special instance of feedforward neural LMs. Given a word w to predict and contexts $c_0, c_1, c_2, ..., c_{n-1}$, the hidden representation of the log-bilinear language model is given by

$$\vec{r} = \sum_{i=1}^{n-1} W_i \vec{v}_{c_i} \tag{4.4}$$

where W_i is a position-dependent weight matrix. By setting all these matrices to the identity, the previous equation simplifies to

$$\vec{r} = \sum_{i=1}^{n-1} \vec{v}_{c_i} \tag{4.5}$$

which is essentially how the context vector is computed in the CBOW algorithm, with the exception that the sum is replaced by an average. In CBOW, as we are only interested in obtaining high-quality embeddings, we may also use as context the words that follow the token to predict in addition to those that precede it. Skip-gram is practically identical, although instead of predicting a word using all its context tokens at once, it does so separately for each of them.

While a softmax could be used to train these models, this would go against the very essence of these methods, which try to obtain embeddings quickly. As an alternative to the softmax, Mikolov et al. use either hierarchical softmax [75] or negative sampling [77], while Mnih and Kavukcuoglu employ the closely related noise-contrastive estimation [81]. In the case of negative sampling, given embedding spaces V and V', to predict a word w with context $c_0, c_1, c_2, ..., c_{n-1}$, the local objective function CBOW maximizes is

$$J = \log\left(\sigma\left(\vec{v}'_w \cdot \sum_{i=0}^{n-1} \vec{v}_{c_i}\right)\right) + \sum_{j=1}^{k} \log\left(\sigma\left(-\vec{v}'_{w^{(j)}} \cdot \sum_{i=0}^{n-1} \vec{v}_{c_i}\right)\right) \tag{4.6}$$

where the words $w^{(j)}$ are randomly drawn negative samples.

It should be noted that although CBOW and Skip-gram have sometimes been presented as deep learning methods, the networks used by these algorithms are actually very shallow. As such, calling these models "deep" is a misnomer.

4.2.2 GloVe

Competitive embeddings may also be learnt using GloVe [91], which stands for Global Vectors. In this case, rather than learning representations by iterating over a large text corpus, a simple predictive model is trained directly on word-context counts. More precisely, given a count matrix M with elements $m_{i,j}$ and a weighting function f, GloVe maximizes

$$J = \sum_{i,j=1}^{|V|} f(m_{i,j})(\vec{v}_i \cdot \vec{v}'_j + b_i + b'_j - \log(m_{i,j}))^2 \tag{4.7}$$

with a variant of stochastic gradient descent. One specific characteristic of the GloVe algorithm is that it is designed as a regression task instead of a classification problem, which lets it sidestep the softmax normalization issue completely.

4.3 Multilingual embeddings

In addition to monolingual data, word vector spaces may also be constructed from bilingual or multilingual resources, which can play a few non-exclusive roles. To leverage the vast quantity of information available in English or other high-resource languages, it may be desirable for the multilingual word embeddings to overlap [57], with a word and its translations close by in a joint vector space, which will allow for some knowledge to be transferred to resource-poor languages. Whether or not there is such overlap, multilinguality may also even improve the English (or other high-resource language) embeddings themselves [31, 48]. Finally, bilingual word embeddings may be learnt to help with a particular task such as machine translation [110] or word sense disambiguation.

4.3.1 Overlapping embeddings

To obtain overlapping multilingual embeddings, a popular approach is to train two language models (or simpler variants) in parallel, while adding a regularization term that will force translations to get close to each other [57, 110]. This regularization term may

be based on word alignments from parallel corpora [86], but simpler variants exist. By representing a sentence as the average of its word embeddings, we may simply minimize the distance square between a source sentence and its translation [41]. This principle has also been applied for learning from parallel data only. While Chandar et al. [18] also use a bag-of-words representation, Hermann and Blunsom [46] propose using slightly more complex composition functions that take word order into account, leading to possibly more expressive models. Yet another alternative is to train two monolingual models completely separately, and then identify a mapping from one space to the other by matching a few known translations [76].

CHAPTER 5

EXPERIMENTS ON WORD ANALOGIES

In the first core chapter of this thesis, we conduct various experiments related to word analogies, a task introduced by Mikolov et al. [78] after observing that many semantic and syntactic relationships could approximately be represented by vector offsets between word embeddings. A now famous example is that when subtracting $\vec{v}(man)$ and adding $\vec{v}(woman)$ to $\vec{v}(king)$, the resulting vector is often very close to $\vec{v}(queen)$.

This chapter is split into two main parts. At first, we consider the usual unsupervised setting, where word representations are learnt on a large corpus of raw text. In particular, we analyze the effects of position-dependent weights and the choice of analogy recovery method. We then switch to a supervised setting, where we present preliminary work on learning from analogies directly.

5.1 Unsupervised setting

5.1.1 Recovering analogies

Given an analogy question of the form $a : b :: c : d$ (read as a is to b as c is to d), where d is unknown, Mikolov et al. [78] have shown that the answer may often be recovered by finding the word vector that has the highest cosine similarity with $(\vec{b} + \vec{c} - \vec{a})$, excluding the representations of a, b and c themselves. Assuming that all embeddings have unit norm, this is equivalent to

$$d^* = \arg \max_{d' \in V \setminus \{a,b,c\}} (\vec{b} \cdot \vec{d'} + \vec{c} \cdot \vec{d'} - \vec{a} \cdot \vec{d'}) \tag{5.1}$$

and we therefore seek the word most similar to b and c and dissimilar from a [67, 82]. Levy and Goldberg [67] have argued that to avoid one word of the analogy question to overshadow the others, it is preferable to switch to a multiplicative combination (3Cos-

Mul)

$$d^* = \arg \max_{d' \in V \setminus \{a,b,c\}} \left(\frac{(\vec{b} \cdot \vec{d'} + 1)(\vec{c} \cdot \vec{d'} + 1)}{(\vec{a} \cdot \vec{d'} + 1) + \varepsilon} \right), \tag{5.2}$$

where ε is a small positive constant to avoid division by zero. Ignoring ε, the previous analogy recovery method may also be written as

$$d^* = \arg \max_{d' \in V \setminus \{a,b,c\}} \Big(\log(\vec{b} \cdot \vec{d'} + 1) + \log(\vec{c} \cdot \vec{d'} + 1) - \log(\vec{a} \cdot \vec{d'} + 1) \Big) \tag{5.3}$$

Hence, if a word is very dissimilar from a, it could get a very high score and be wrongly identified as the answer to the analogy question. While many analogy recovery methods could possibly be used to address this issue, we will only experiment with

$$d^* = \arg \max_{d' \in V \setminus \{a,b,c\}} \Big(\tanh(\vec{b} \cdot \vec{d'}) + \tanh(\vec{c} \cdot \vec{d'}) - \tanh(\vec{a} \cdot \vec{d'}) \Big) \tag{5.4}$$

To see why this may work better, looking at figure 5.1 is useful. For high similaries (>0.5), both non-linearities behave very similarly, but the *tanh* doesn't explode for very small values, contrarily to the logarithm.

5.1.2 Position-dependent weights and model combination

Many word representation algorithms may be used to obtain vectors suitable for word analogies or other NLP tasks. In this chapter, we mostly focus on CBOW vectors trained with negative sampling. We also consider a variant with position-dependent weights [82] where given a token w_i to predict, a $2m$-word bidirectional context and learnt weights $\vec{z}_n$ [1], the context vector is given by $\sum_{\substack{-m \leq n \leq m \\ n \neq 0}} \frac{\vec{z}_n \odot \vec{w}_{i+n}}{2m}$ instead of $\sum_{\substack{-m \leq n \leq m \\ n \neq 0}} \frac{\vec{w}_{i+n}}{2m}$. In particular, such a model may be seen as a log-bilinear LM with diagonal weight matrices (see eq.4.4). While worst performance was reported with similar vectors in [82], there have since been positive reports [2].

In most analogy questions we consider in this chapter, words b and c usually play a

1. We had experimented with fixed masks instead of position-dependent weights, but results were much worse.

2. https://groups.google.com/forum/#!topic/word2vec-toolkit/jdT6JX3H_8k
https://groups.google.com/forum/#!topic/word2vec-toolkit/f3DGMy8RRe4

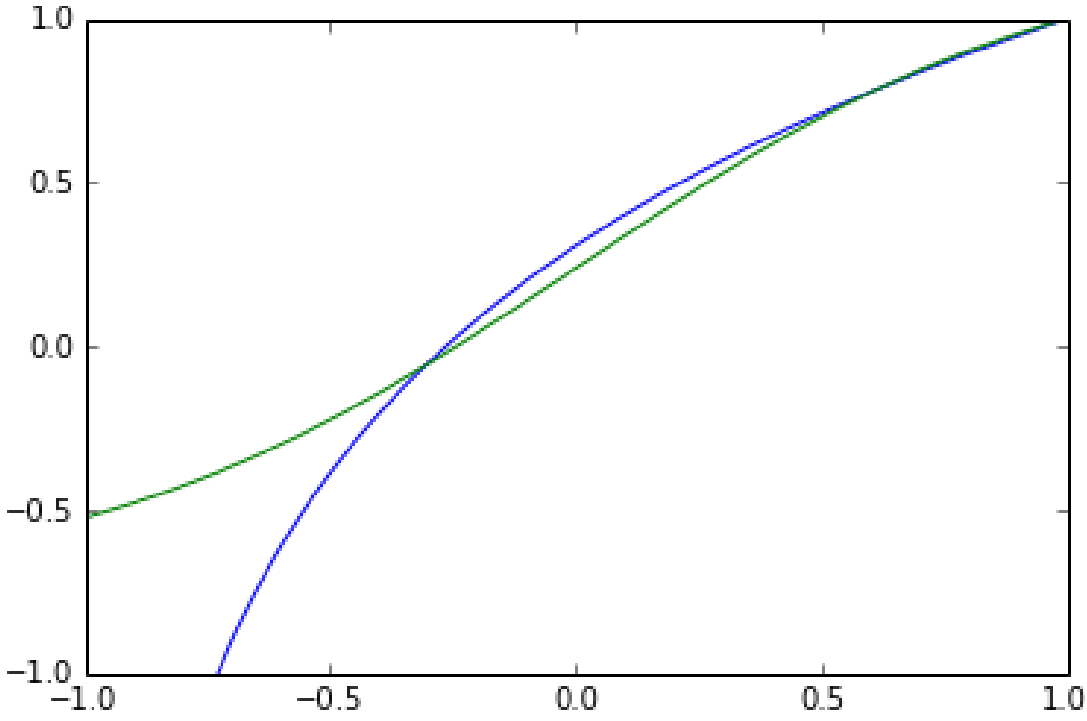

Figure 5.1: Visualization of $\log(x+1)$ (in blue) and $\tanh(x)$ (in green). Both functions are shifted so that their value at 1 is 1.

different role as for the analogy to be valid, there will generally be a clearly defined transformation between a and b as well as c and d, but not necessarily between a and c or b and d, which will often be hyponyms (members of a more general category). Turney [101] argues that pairs ab, cd have high domain similarity whereas ac, bd are functionally similar. For example, in semantic analogies such as *Athens* : *Greece* :: *Baghdad* : *Iraq*, *Baghdad* and *Iraq* are both in the domain of Iraqi "things", while *Greece* and *Iraq* share the same function as countries. In the context of syntactic analogies, functional similarity should probably be understood as syntactic similarity, which should be high for words that have the same fine-grained part of speech. One hypothesis we wish to verify (or infirm) in this chapter is whether the usual bag-of-words are better suited to domain similarity while position-dependent weights would help capture functional or syntactic similarity. To do so, we consider a few basic model combination functions. In the first

case (Forward), we may use

$$d^* = \arg\max_{d' \in V \setminus \{a,b,c\}} \left(f(\vec{b}_{PDW} \cdot \vec{d'}_{PDW}) + f(\vec{c}_{BOW} \cdot \vec{d'}_{BOW}) - \tfrac{1}{2}(f(\vec{a}_{BOW} \cdot \vec{d'}_{BOW}) + f(\vec{a}_{PDW} \cdot \vec{d'}_{PDW})) \right) \tag{5.5}$$

where f is a non-linearity such as those discussed is the previous section, and $\vec{w}_{PDW}$ and $\vec{w}_{BOW}$ are respectively the representations of word w that were learnt with and without position-dependent weights. We may contrast this by interchanging the roles of the vector spaces (Backward), which we would expect to perform worse if the hypothesis we formulated is indeed true.

$$d^* = \arg\max_{d' \in V \setminus \{a,b,c\}} \left(f(\vec{b}_{BOW} \cdot \vec{d'}_{BOW}) + f(\vec{c}_{PDW} \cdot \vec{d'}_{PDW}) - \tfrac{1}{2}(f(\vec{a}_{BOW} \cdot \vec{d'}_{BOW}) + f(\vec{a}_{PDW} \cdot \vec{d'}_{PDW})) \right) \tag{5.6}$$

Finally, we also consider the simple average of the scores given by the two models.

$$d^* = \arg\max_{d' \in V \setminus \{a,b,c\}} \sum_{M \in \{BOW,PDW\}} \frac{1}{2} \left(f(\vec{b}_M \cdot \vec{d'}_M) + f(\vec{c}_M \cdot \vec{d'}_M) - f(\vec{a}_M \cdot \vec{d'}_M) \right) \tag{5.7}$$

While these combination methods are quite crude, and that ideally we may want to learn the weights associated to each term [3], we shall nevertheless soon see that some valuable insight may be gained from these experiments.

5.1.3 Experimental details

To train the different models, we mostly follow the settings of the *big model* script [4] of the *word2vec* software, which uses various online sources [5] to build a corpus of approximately 8 billion words, after which short phrases are formed [77]. Using a custom version of the Gensim library [92], we train 500-dimensional vectors over 3 epochs, using a window of 10 words on both sides of the target, 10 negative samples and a subsampling threshold of 10^{-5} [77]. To reduce memory usage, all words occuring less than

3. Using cross-validation instead should be possible, and maybe desirable, but it would be somewhat tricky as many analogies share words in common
4. http://word2vec.googlecode.com/svn/trunk/demo-train-big-model-v1.sh
5. We use an older version of the Wikipedia dump (March 2014)

50 times are discarded. Moreover, to accelerate training, we use 8 threads even though this makes results non-deterministic. For the standard CBOW model, we use a learning rate that decreases linearly from 0.05 to zero. For the model with position-dependent weights, following a suggestion by Tomas Mikolov [6], each thread has its own weights. In this case, to avoid numerical instabilities caused by this change, the initial learning rate is reduced to 0.025.

To evaluate the models, two analogy datasets are used. The first one, released by Microsoft Research (MSR), contains 8000 syntactic questions spanning 8 categories. The second one, from Google, has 19,544 analogies. Some of them are similar to those in the MSR data, while others have a more semantic nature. Examples for each type of analogy considered are presented in Appendix I. To hasten the evaluation of the models, we only consider the most frequent 400,000 words instead of the whole vocabulary. Moreover, given the tokenization of the training corpus, one category of the MSR dataset, possessive nouns (city:city's), is discarded. Because of these restrictions, we report results for 19,486 Google and 6964 MSR questions.

5.1.4 Results

For single models, Table 5.I presents detailed results on both analogy tasks. For the Google dataset, we confirm previous reports that announced benefits with position-dependent weights. This effect is even more pronounced on the MSR dataset, with overall gains of more than 10%, although a more thorough hyper-parameter search could possibly reduce that gap [69].

For both models and both datasets, 3CosMul indeed works better than 3CosAdd, reinforcing the claims of [67, 69]. Nevertheless, 3CosTanh surpasses both of these analogy recovery methods, albeit marginally. In any case, we believe that there almost surely is an infinity of non-linearities that could outperform the methods we have evaluated. Moreover, while the model with position-dependent weights evaluated with the tanh non-linearity generally does better than the others, it is still beaten on some analogy categories.

6. https://groups.google.com/forum/#!topic/word2vec-toolkit/jdT6JX3H_8k

Google	BOW			PDW		
	x	$\log(x+1)$	$\tanh(x)$	x	$\log(x+1)$	$\tanh(x)$
capital-common-countries	96.0	95.7	95.7	98.4	98.4	**98.6**
capital-world	94.1	93.6	93.8	93.4	94.1	**94.2**
currency	20.3	21.9	**22.3**	18.1	20.0	19.8
city-in-state	83.0	83.3	**83.5**	75.0	77.7	78.5
family	87.4	88.3	87.5	91.1	92.3	**92.5**
gram1-adjective-to-adverb	36.9	35.2	**37.3**	25.0	34.2	34.8
gram2-opposite	37.6	35.2	37.1	41.0	42.0	**42.1**
gram3-comparative	88.2	85.8	86.6	**94.7**	93.9	94.4
gram4-superlative	87.3	91.1	92.4	90.1	93.9	**94.2**
gram5-present-participle	74.3	75.3	75.6	81.6	84.1	**84.2**
gram6-nationality-adjective	89.6	89.2	89.3	90.0	90.4	**90.5**
gram7-past-tense	72.4	74.5	76.2	78.1	**83.0**	82.8
gram8-plural	88.3	88.1	88.7	91.5	94.1	**94.6**
gram9-plural-verbs	61.3	67.7	67.4	84.5	89.5	**89.9**
total	78.4	78.7	79.3	79.6	81.8	**82.1**

MSR	BOW			PDW		
	x	$\log(x+1)$	$\tanh(x)$	x	$\log(x+1)$	$\tanh(x)$
adj. base/comparative	65.7	67.8	69.3	74.1	76.6	**77.0**
adj. comparative/superlative	66.4	68.1	68.4	75.1	77.0	**77.2**
adj. base/superlative	65.5	69.0	70.0	72.6	76.4	**76.9**
nouns singular/plural	61.5	54.7	56.5	67.4	70.2	**70.9**
verbs base/past	61.6	63.6	66.2	80.0	82.9	**83.5**
verbs past/third person	61.9	63.2	65.6	79.4	82.3	**82.6**
verbs base/third person	55.7	59.1	61.3	78.7	84.7	**85.2**
total	62.6	63.6	65.3	75.3	78.6	**79.0**

Table 5.I: Single-model results (in %) on analogy tasks. BOW uses a standard bag-of-words context, while PDW adds position-dependent weights.

When averaging the different models according to the *Backward* and *Forward* methods, some interesting patterns appear. Although the *Forward* and *Backward* methods only differ in the role each embedding space plays, there is a significant gap in performance, with *Forward* having the upper hand on 18 of the 21 sub-categories. In particular, this gap is surprisingly big on some types of analogy. For example, in the Google dataset, the absolute difference between both strategies is 17.8% and 34.4% respectively for *city-in-state* and *gram1-adjective-to-adverb*. However, it is important to note that these two

combination heuristics are not particularly great since they are very often superseded by either the best model or by the *Average* of the scores. Nevertheless, we believe that some insight is gained from these mostly negative results.

Google	Backward	Forward	Average
capital-common-countries	91.5	**99.8**	97.8
capital-world	85.5	**96.4**	95.7
currency	**22.6**	18.9	22.5
city-in-state	69.4	**87.2**	83.9
family	90.7	81.0	**92.1**
gram1-adjective-to-adverb	10.0	**44.4**	40.7
gram2-opposite	36.7	40.1	**45.2**
gram3-comparative	90.1	88.1	**92.8**
gram4-superlative	88.1	94.3	**95.3**
gram5-present-participle	65.2	82.2	**84.1**
gram6-nationality-adjective	78.4	**90.2**	**90.2**
gram7-past-tense	66.7	78.7	**83.3**
gram8-plural	79.7	92.6	**94.5**
gram9-plural-verbs	71.1	71.4	**83.6**
total	71.5	82.0	**83.3**

MSR	Backward	Forward	Average
adj. base/comparative	65.8	72.5	**76.9**
adj. comparative/superlative	68.2	73.4	**76.0**
adj. base/superlative	68.3	73.3	**77.9**
nouns singular/plural	47.4	64.2	**69.5**
verbs base/past	64.0	67.0	**81.4**
verbs past/third person	61.9	66.3	**78.7**
verbs base/third person	61.8	63.4	**81.2**
total	62.4	68.6	**77.4**

Table 5.II: Model combination results (in %) on analogy tasks. In all cases, the tanh non-linearity is used.

Although the *Forward* combination method is not as effective as we might have expected, there is nevertheless some other evidence that supports the hypothesis that position-dependent weights help capture functional or syntactic similarity while simple bag-of-words are appropriate for domain similarity (relatedness). In particular, for the analogy questions we considered, we may compute the average cosine similarity

between the expected answers d and the corresponding question words a, b and c. Table 5.III summarizes the results on each dataset. For a more fine-grained portrait, refer to Appendix II. For the Google dataset, unrelated words (a) have comparable cosine similarity to d. However, the bd similarity is on average 0.09 greater for the BOW model, while the opposite happens for cd, with the PDW model now holding an advantage of 0.08. On the MSR dataset, it is more difficult to draw definitive conclusions as the unrelated words a have fairly different cosine similarities with d [7].

	a		b		c	
	BOW	PDW	BOW	PDW	BOW	PDW
Google	0.17	0.19	0.31	0.40	0.64	0.56
MSR	0.08	0.15	0.24	0.31	0.56	0.57

Table 5.III: For word analogy questions of the form $a : b :: c : d$, average cosine similarity of question words a, b, c with the expected answer d, either without (BOW) or with (PDW) position-dependent weights

5.2 Supervised learning of analogies

Until now, by using no analogy training or validation set, we had our hands quite tied as we could not conduct many experiments without overfitting to the test sets. In this section, we propose learning about analogies directly on the large WordRep dataset [38], which contains millions of questions. While this dataset was previously used to embed relations and recover analogies [108] in a supervised manner, their learning criterion only involved word pairs instead of complete analogy questions.

5.2.1 Training, validation and test data

As test data, we use the Google analogy dataset discussed in the previous sections and build the training and validation sets from WordRep [38]. To correctly estimate the ability of the embeddings to generalize to new analogies, we make sure that any word may only appear in at most one of these three sets. Because of this constraint, there are

7. The absolute cosine similarities matter much less than relative similarities. If two models had the same pairwise cosine similarities up to some fixed offset, they would be very similar to each other.

a few categories, such as "capital-world" or "city-in-state", for which very little training data may be gathered. For this reason, we consider only syntactic analogies for training and validation, but will report test results for both kinds of analogies.

For each of the 9 types of syntactic analogies covered in the syntactic section of test set, if there are at least 80 available word pairs after filtering the test set out, we set aside 40 for validation. Otherwise, we use half of the available data for validation and the other half for training. While most categories have more than 500 correct training/validation pairs, "gram3-comparative", "gram4-superlative" and "gram6-nationality-adjective" stand out with only 51, 44 and 20.

We then create the analogy validation set by building $10,000$ random valid questions from the word pairs, with an approximately uniform distribution over the different categories.

5.2.2 Model and learning algorithm

To hopefully obtain embeddings that are better at solving analogy questions, we simply try to learn a linear transformation of the embedding space, replacing the embedding matrix E by EW.

Given an analogy question $a : b :: c : d$, we first compute the score s of every possible answer using equation 5.4 since this was the most effective technique we previously found in the unsupervised setting. After normalizing all scores to mean 0 and standard deviation 1, we then minimize a hinge-like loss function [24] [8] for every training analogy case.

$$J = \log\left(1 + \min\left(0, 1 - s(d) + \max_{d' \notin \{a,b,c,d\}} s(d')\right)\right) \tag{5.8}$$

As such, when the model correctly predicts the answer to an analogy question, the value inside the hinge will be less than 1 and vice-versa.

With a constant learning rate of 0.1 and early-stopping on the validation set, we trained the model over 400 mini-batches of 1000 analogy questions. Even though the loss function is quite inefficient, learning took approximately one hour, which we deem

8. It may be possible to use a simpler loss function, but this is the first one that worked decently well.

acceptable.

5.2.3 Results

On the validation set, the accuracy of the position-dependent model was initially 82.22% and rose up to 88.07% during training. Although we normalized the matrix EW before every time step so that it had unit-norm rows, the best results on the development set were obtained without such normalization. As such, to compute the final results on the test set, we used EW directly.

Google	Original	Transformed
capital-common-countries	**98.6**	97.0
capital-world	**94.2**	92.5
currency	19.8	**21.4**
city-in-state	**78.5**	74.0
family	**92.5**	90.3
gram1-adjective-to-adverb	34.8	**50.8**
gram2-opposite	**42.1**	33.3
gram3-comparative	**94.4**	85.4
gram4-superlative	**94.2**	88.7
gram5-present-participle	84.2	**91.0**
gram6-nationality-adjective	**90.5**	89.1
gram7-past-tense	82.8	**90.0**
gram8-plural	94.6	**95.5**
gram9-plural-verbs	89.9	**92.9**
total	**82.1**	81.6

Table 5.IV: Word analogy results (in %) with and without supervised learning

On the semantic analogy questions, for which there was no training data, the transformed vectors generally do worse than the original ones, while results are a bit mixed for syntactic questions. Out of the 4 syntactic categories for which the original vectors are better, 3 had much less training data available, which may help explain why learning was not successful in these cases. As for the other problematic category, "gram2-opposite", it may be that the training and validation sets are poor proxies for the test set. Indeed, in the WordRep database, all syntactic categories of the Google dataset are mapped to another of the same name, except for "gram2-opposite" which is merged into

a more general “Antonym” subtask. Nevertheless, for most kinds of syntactic analogies on which data is plentiful, supervised learning does help.

5.3 Summary

In this section, we have first confirmed previous findings about the usefulness of position-dependent weights for answering analogy questions. We have also shown that current techniques to recover analogies are sub-optimal. Even if the model combination heuristics we considered have been only very mildly successful, these experiments let us gain some insight into the behaviour of word embeddings. Finally, we have presented an approach that allows learning on analogies directly. While gains were often observed for subtasks with plentiful data, the accuracy on categories for which we used little or no training data worsened.

CHAPTER 6

SIMPLE BILINGUAL EMBEDDINGS FOR WORD SIMILARITY

In this chapter of this thesis, we extend the work of Hill et al. [47, 48] on understanding the emergence of word similarity from bilingual embeddings. Using very simple shallow neural embeddings with bilingual contexts, we are able to match the performance of neural machine translation embeddings on two similarity tasks. We also investigate the use of various parallel corpora spanning multiple languages and find that one of them, despite being shorter than some others, allows to get significantly better results, matching the state of the art on the TOEFL test and approaching human-level performance on the Simlex-999 word similarity gold standard. Finally, we show that, while the generated English and foreign embeddings don't naturally overlap, we may do so without degrading significantly the quality of the vector spaces.

6.1 Word similarity

Informally, *similarity* measures, on a continuous spectrum, the degree to which two words are synonymous. As such, antonyms or completely unrelated words are very dissimilar to each other, while other word pairs, such as *pool* and *ocean*, which have some attributes in common, but are clearly not synonymous, would be moderately similar. Similarity differs from *association*, or *relatedness*, which measures the strength of the relationship between two words [49]. While two synonyms are generally highly related to each other, so are antonymous words. Aside from synonyms and antonyms, other examples of strongly related words include *wheel-car* or *baby-cries*. Conversely, most people would consider that *lightning* and *blueberries* are not closely related even though it is possible, with some effort, to find a relationship between these two words [1].

To capture both relatedness and similarity, traditional approaches often rely on the distributional hypothesis [33, 44], which states that words that share similar contexts

1. The lightning may cause a forest fire that will help blueberry growth.

have similar meanings. In this chapter, we consider the multilingual version of that hypothesis, namely that words or phrases that have translations in common will often be highly similar, if not synonymous [4, 48, 104, 107].

6.2 Models

To create word representations that capture similarity well, we use the skip-gram or CBOW embeddings described in section 4.2.1 [2], where rather than using nearby words or dependency links [66] to build the contexts, we rely on bilingual knowledge. More precisely, after obtaining word alignments from a word-based translation system [86], the model must predict, for each aligned source word, what its translation is. Such contexts were previously used by van der Plas and Tiedemann [104] and, as such, the vectors we build are a simple prediction-based variant of their VSMs.

With these alignment-based contexts, words that share common translations should be pulled nearby each other in the embedding space. While such direct interactions may suffice to capture synonymy, we may also expect the model to learn a continuous similarity spectrum indirectly. For example, if there are two source words s_0 and s_1 that are always or often aligned to t_0 and t_1 respectively, and a third one s_2 that may be translated by either one of these, then we hypothesize that s_0 and s_1 should be brought closer to each other than to most other words.

6.2.1 Superposing source and target embeddings

A common objective of many bilingual or multilingual word representation algorithms is to make the embeddings of source words and their translations close to each other [18, 41, 45, 46, 56, 61]. Given that we train our vectors with negative sampling, some could intuitively think that such a property would be observed here, but this is not the case. If such a constraint on the embeddings is desirable, we could try adding a regularization term that minimizes the distance squared between each source-target pair [41], but preliminary experiments with such a change were disappointing. To obtain

2. These models are identical if the context is a single word.

overlapping vector spaces, this L2 regularization term had to be weighted so heavily that the underlying embeddings were rendered practically useless. To sidestep this issue, we propose using two embedding spaces for each language and use cross-model L2 regularization. Given representations e, E on the source side and f, F for the target, we may maximize the per-token objective

$$\begin{aligned} J_i = &\log\left(\sigma\left(\vec{e}_i \cdot \vec{f}_i^{(0)}\right)\right) + \sum_{j=1}^{k} \log\left(\sigma\left(-\vec{e}_i \cdot \vec{f}_i^{(j)}\right)\right) \\ &+ \log\left(\sigma\left(\vec{F}_i \cdot \vec{E}_i^{(0)}\right)\right) + \sum_{j'=1}^{k} \log\left(\sigma\left(-\vec{F}_i \cdot \vec{E}_i^{(j')}\right)\right) \\ &+ \frac{\gamma}{2}\left(||\vec{e}_i - \vec{F}_i||^2 + ||\vec{E}_i^{(0)} - \vec{f}_i^{(0)}||^2\right) \end{aligned} \tag{6.1}$$

where $\vec{f}_i^{(j)}$, $\vec{E}_i^{(j')}$ are noise samples. With $\gamma = 0$, the above objective would let us learn two bilingual skip-gram models independently. By increasing γ, the source embeddings of one model will align with the target representations of the other and vice-versa.

6.3 Experimental procedure

To evaluate how well these models can capture similarity, we use two tasks, the TOEFL synonym test [64] and the SimLex-999 gold standard [49]. The first one is a 80-question multiple choice test introduced to the NLP community by Landauer and Dumais where, given a stem word, one must choose the correct synonym amongst four candidates. To do so, we simply pick the word with the highest cosine similarity with the stem. Somewhat unfortunately for our purpose, the distractors are not always closely related to the stem word, which may let models that don't distinguish relatedness from similarity particularly well have a decent score nevertheless. Given the small number of questions and the relatively high amount of rare words amongst the stems and candidates [35], it is also unclear how performance on this task generalizes to others.

On SimLex-999, the task is to rank 999 word pairs from least to most similar, where we once again use cosine similarity. One defining characteristic of this dataset, contrary to other gold standards such as MEN[14], is that dissimilar but related words such as

night and day are given a relatively low score. We also report results on the Simlex-333 subset, which contains the 333 most highly associated pairs but nevertheless covers the full similarity range. As such, models that struggle to disentangle relatedness and similarity are expected to perform even more poorly on this subset.

6.3.1 Hyper-parameters

Unfortunately, these two datasets do not have a standard validation-test split, but the skip-gram algorithm has some hyper-parameters that could affect performance. To keep the computational effort low and avoid overfitting the data too much, most settings are fixed to reasonable default values. As such, we use vectors of dimension 500, a minimum word count of 5, 10 negative samples from the unigram to the $\frac{3}{4}th$ power, and a learning rate that linearly decreases from 0.05 to approximately 0 at the end of training.

Nevertheless, as the impact of some hyper-parameters is not clear, we modify some of them. To give a fair depiction of the performance, we will report scores for most of the important settings. Even though Levy et al. [69] showed that gains from hyper-parameter tuning would often generalize well, the very best numbers we report should be seen as optimistic estimates of the true performance.

6.3.1.1 Data conditions

As the quality of word embeddings depends on the resources needed to create them, we explore the use of different corpora. At first, to see whether some languages are more effective than others at inducing (English) similarity, we use a common set of 479,260 tokenized lowercased English sentences from the Europarl corpus for which there is an available translation in Czech (Cs), German (De), Spanish (Es), Finnish (Fi) and French (Fr). While larger datasets may have been desirable, the amount of sentences translated into many languages is quite limited. Second, to allow for fair comparison with Hill et al. [47, 48], we replicate their experimental conditions for both English-French and English-German, using corpora from WMT'14 [3]. For this data condition, the models

3. See section 8.5 for more details

are evaluated on the same subset of 886 SimLex-999 pairs and 55 TOEFL questions. [4]. Finally, for all previously mentioned language pairs, as well as English-Russian (Ru), we use all data made available by the Tenth Workshop on Statistical Machine Translation (WMT'15). In this case, in addition to lowercasing and tokenization, we also clean the corpora by removing sentences of length greater than 80, source sentences that are at least two times plus five longer than the corresponding target sentences and vice-versa, and sentences that are not written in the correct language according to an external language detection toolkit [95]. Generally, models are trained over 5 epochs, except for the first data condition for which we use 10 due do the small size of the training corpora.

6.3.1.2 Alignment

Generally, to obtain word alignments, two unidirectional models are constructed, with one that tries to align each English word to a single foreign word (or to a special *NULL* token) and vice-versa. To accomplish this step, we use the *fast_align* tool[30]. The two resulting models may then be combined, or symmetrized, using a variety of techniques [59]. From the most total alignment points to the least, we try the *union*, *grow-diag-final*, *grow-diag-final-and*, *grow-diag* and *intersection* heuristics. In general, using less alignment points will lead to higher precision (few erroneous alignments) at the cost of a lower recall (more good alignment points missed), but the impact on similarity tasks is not immediately clear. By default, we use the *intersection* of the two models.

6.3.1.3 Subsampling

To both accelerate training and potentially obtain better results, Mikolov et al. [77] subsample words according to their frequency. More precisely, given a threshold t and the total number of words C (with repetitions), a word with count c remains in the corpus with probability

$$p = \min\left(\sqrt{\frac{Ct}{c}} + \frac{Ct}{c}, 1\right) \tag{6.2}$$

4. For English-French, we did restrict the corpus to sentences of length 50 or less.

We either use no subsampling or, by default, a threshold of $t = 10^{-5}$ for both languages. The latter is quite aggressive since a training pair will be discarded if either the source or target word is deleted.

6.3.1.4 Regularization

Although we mostly train unidirectional models, with English as the source and foreign languages as target, we also examine the effect of γ for the bidirectional models described in subsection 6.2.1. In addition to quantitative evaluation, we qualitatively observe how the different vector spaces interact with each other.

6.4 Results

Table 6.I presents the performance, with default hyper-parameters, of the bilingual skip-gram (Bil-SG) when using 479k English sentences and their translations into multiple languages. On both tests, while all languages are effective to some degree, it seems harder to model similarity using the Finnish corpus, possibly because of its very large vocabulary and the high number of low-frequency tokens.

	Cs	De	Es	Fi	Fr
TOEFL (%)	80.0	87.3	80.0	69.1	80.0
SimLex-999 (ρ)	0.26	0.25	0.29	0.20	0.30

Table 6.I: Performance on TOEFL and SimLex-999 for small bilingual corpora (default hyper-parameters). Results are given on the 55 TOEFL questions and 772 SimLex-999 pairs covered by the English vocabulary

As reported in Table 6.II, on the corpora used by Hill et al. [48], performance on both tasks improves significantly, matching or slightly improving upon that of NMT systems[48]. Given the streamlined approach taken here, these results clearly further strengthen the multilingual distributional hypothesis.

When using all data available for WMT'15, the different bilingual models often reach more than 90% accuracy on the TOEFL test and correlations over 0.5 on the SimLex-999 gold standard (Table 6.III). On both of these tasks, the worst model is English-Finnish, which is not particularly surprising given that there is little training data and

	NMT		Bil-SG	
	De	Fr	De	Fr
TOEFL (%)	98.2	92.7	100.0	94.5
SimLex-999 (ρ)	0.50	0.53	0.52	0.57
SimLex-333 (ρ)	0.47	0.49	0.55	0.59

Table 6.II: Comparison with the best results obtained with NMT embeddings on the 55 TOEFL questions, 886 SimLex-999 and 291 SimLex-333 pairs used in [48]. (Default hyper-parameters)

that this is the language pair for which performance was lowest in the first data condition we considered. On the other end of the spectrum, the English-Czech models perform better than all others, despite using less data than both the English-Spanish and English-French systems. Although we cannot provide a definitive answer as to why learning from Czech works so well, we hypothesize that the diversity of the CzEng corpus is helpful. Indeed, in opposition to many parallel corpora, which often contain news articles or parliamentary proceedings, CzEng also covers movie subtitles, fiction, etc.

	Cs	De	Es	Fi	Fr	Ru	CBOW	PDW	SOTA
Corpus length (M)	416	201	778	87	2083	87	7644	7644	-
TOEFL (%)	97.5	92.5	93.8	76.3	95.0	87.5	91.25	91.25	100.0
SimLex-999 (ρ)	0.65	0.57	0.57	0.31	0.60	0.55	0.38	0.47	0.64
SimLex-333 (ρ)	0.65	0.59	0.58	0.40	0.55	0.61	0.18	0.30	0.49

Table 6.III: Performance on TOEFL, SimLex-999 and SimLex-333 with all WMT'15 data (default hyper-parameters). Results on the TOEFL tasks are reported over all 80 questions, even if there are out-of-vocabulary words. For SimLex-999, there are 998 valid pairs for Cs, De, Es, 997 for Ru and 941 for Fi. We also present results for the embeddings used in chapter 5 and for state-of-the-art models [3, 16, 48] For reference, the size of the different corpora is mentioned, counting both source and target words.

If we compare them against state-of-the-art systems, the bilingual embeddings fare quite well. For SimLex-999, the correlation is marginally greater than the best previous result [3], which was however obtained by combining multiple models. On TOEFL, the bilingual embeddings do not yet reach the perfect accuracy of Bullinaria and Levy [16], but we still have not tuned hyper-parameters as they did.

6.4.1 Impact of hyper-parameters

Most models seem reasonably robust to changes in the alignment heuristic. On SimLex-999, the difference between the best and worst correlation is less than 0.05 for all language pairs (except Finnish-English), with no heuristic systematically better than the others. The best overall score we obtained was 0.66, which is very close to human inter-agreement (0.67) [49]. On the subset of 333 most associated pairs, using the intersection of the two alignments often works best, possibly because related but dissimilar words are less often wrongly aligned.

There is little variation on the TOEFL task, with a difference of at most 4 correct answers out of 80 for any language pair. Nevertheless, for English-Czech, we attain perfect 100% accuracy with many different heuristics. This is quite impressive as, compared to the monolingual models that also reach such levels of performance [16], we used about five times less data (counting both source and target words) and, arguably, explored a much smaller set of hyper-parameter settings.

With the intersection heuristic, if we do not subsample source and target words, performance on SimLex-999 and the TOEFL test almost always degrades. In the worst case over multiple language pairs, we observed a deterioration of 0.07 points and 3.75% on these two tasks. These results are in agreement with the observations of Gouws et al. [41] on a different bilingual variant of skip-gram (BilBOWA), where subsampling led to "better-looking" embeddings.

With the proposed changes in subsection 6.2.1, the quality of the embeddings does not seem affected too much, either positively or negatively, when using moderate amounts of regularization. More precisely, with the WMT'14 English-French data [5], the SimLex-999 scores worsen by at most 0.04 for multiple values of γ inferior to 1, but the accuracy on TOEFL sometimes goes up by a few percentage points. Unsurprisingly, when the regularization is too strong, performance starts to decrease quickly. For example, with $\gamma = 10$ or 100, the SimLex-999 correlations drop to 0.42 and 0.13, although using a smaller learning rate may possibly help.

5. We now evaluate with the full vocabulary.

6.5 Visualization

Figure 6.1 depicts the bilingual vector spaces learned with negative sampling. As announced previously, although it may appear that negative sampling should bring translations nearby each other, the space is actually segmented into two large regions, with little overlap between English and foreign words. While we did not verify this thoroughly, we believe that a similar phenomenon should happen when using negative sampling CBOW or skip-gram with bag-of-words or dependency-based contexts [6]. [66].

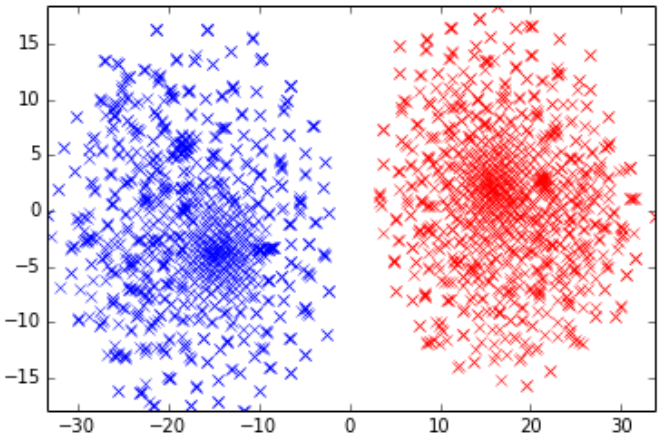

Figure 6.1: t-SNE visualization [103] of the 1000 most common English and French words when learning bilingual embeddings with negative sampling. English words are red, while French ones are blue.

For $\gamma = 0.1$, the four learnt vector spaces have a very clear structure (figure 6.2), with each of the two English vector spaces overlapping a French one. As such, it appear that the regularization term indeed works as intended. To verify this furthermore, we present a small region of the embedding spaces in figure 6.3, where we may see that English words and their translations are often very close to each other.

6. Obviously, if one chooses to share a single matrix for both representations, which is possible when using bag-of-words, the observed phenomenon cannot happen.

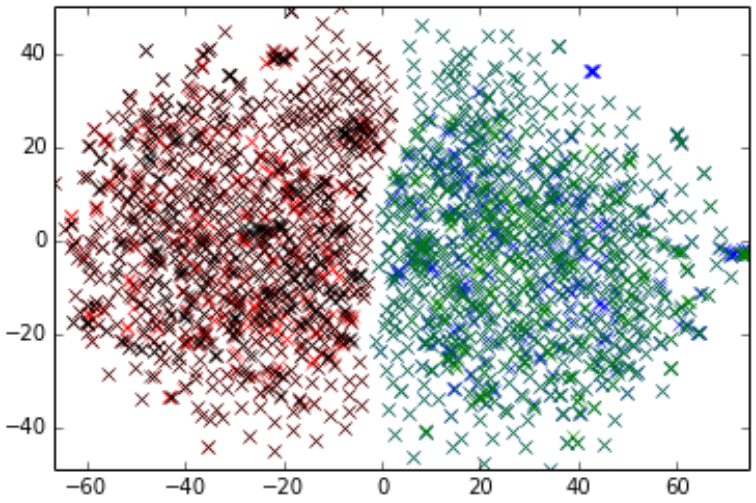

Figure 6.2: t-SNE visualization [103] of the 1000 most common English and French words when learning bidirectional bilingual embeddings with negative sampling ($\gamma = 0.1$). English words are red and green, while French ones are blue and black. Note that each word has two distinct representations.

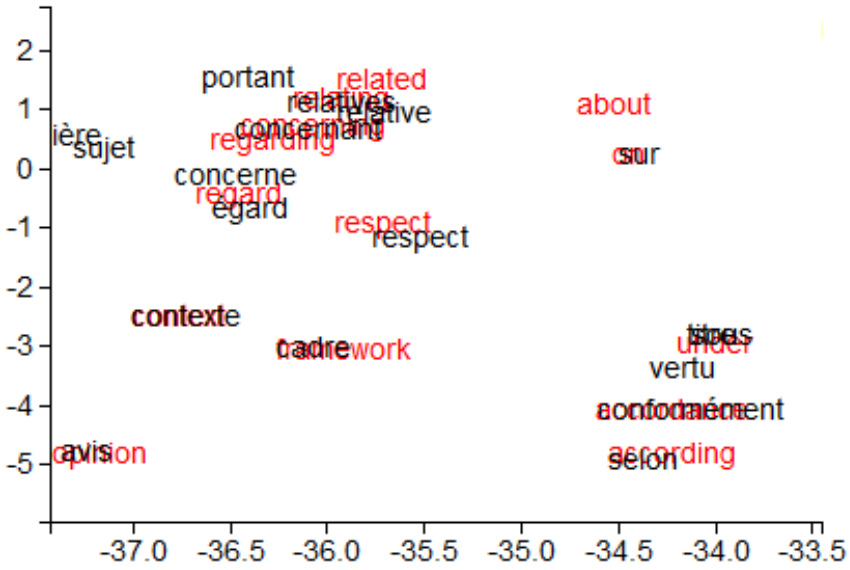

Figure 6.3: t-SNE visualization [103] of a few English and French words when learning bidirectional bilingual embeddings with negative sampling ($\gamma = 0.1$).

6.6 Related work

Bilingual or multilingual information has previously been used to improve word embeddings, for example using canonical correlation analysis (CCA) [31]. Moreover, Faruqui et al. [32] retrofitted (adapted) multiple existing word vectors to the paraphrase database (PPDB) [37], which was created using word alignments to discover synonyms and paraphrases.

Bilingual knowledge has also been employed with the skip-gram or CBOW algorithm in slightly different ways. In addition to BilBOWA [41], Wolf et al. [106] use word alignments, but instead of choosing the aligned word as context, they take its neighbours. As for Garcia et al [39], to help with word sense disambiguation, they create a corpus of aligned source-target words $((s_1,t_1),(s_2,t_2),(s_3,t_3),(s_4,t_4),...)$ and try to predict (s_i,t_i) using nearby word pairs as context.

6.7 Summary

In this chapter, we have shown that very simple bilingual models could reach or surpass the state of the art on both the SimLex-999 and TOEFL tasks. Moreover, when desired, the representation algorithm may be altered to let the source and target embeddings overlap, a phenomenon that doesn't occur naturally when learning with negative sampling. We have also highlighted the particularly good suitability of the WMT'15 English-Czech corpus for inducing word similarity.

Even though we have obtained good results on two similarity tasks, we must caution the reader that the bilingual vectors discussed in this chapter are not necessarily ideal in all cases. For example, on the word analogy task described in the previous chapter, we obtained much worse results than with either monolingual or NMT embeddings.

CHAPTER 7

PROLOGUE TO ARTICLE ON LARGE VOCABULARY NEURAL MACHINE TRANSLATION

7.1 Article details

On Using Very Large Target Vocabulary for Neural Machine Translation
Sébastien Jean, Kyunghyun Cho, Roland Memisevic and Yoshua Bengio
Accepted at ACL-IJCNLP 2015

7.2 Motivation

Previously to the work presented in chapter 6, Hill et al. [47] had observed that embeddings obtained from neural machine translation systems [2, 21] could answer syntactic analogy questions and capture similarity very well despite being trained on relatively little data. However, the usefulness of these embeddings was unfortunately somewhat limited as the vocabulary size had to be restricted to only $30,000$ words. While existing speed-up techniques could possibly have been used to simply obtain large vocabulary NMT embeddings, GPU memory usage was a serious issue to us, especially on the 3 GB GTX 780 Ti the lab had provided us. Faced with this issue, we developed efficient low-memory techniques to train NMT systems.

As reported in [48], we succeeded in building large-vocabulary embeddings. Moreover, although this was somewhat surprising at the time given the approximations we made, the performance on actual translation tasks was quite good. However, even if training was reasonably efficient, translating new sentences was still taking much time. This motivated us to continue working on this topic and to extend the previous ideas to word generation, leading to practical large-vocabulary NMT.

7.3 Individual contributions

I came up with most of the main ideas presented in the article, implemented them and executed all experiments. The article itself was written in collaboration with Kyunghyun Cho, who also supervised my progress and encouraged me in pursuing this research. Both before and during the redaction, numerous discussions with Roland Memisevic helped improve the quality of the paper. Moreover, Yoshua Bengio established the relationship between the proposed training approach and importance sampling.

I am also thankful to Dzmitry Bahdanau, who contributed greatly to the development of efficient neural machine translation [2], to Felix Hill for his work on word embeddings [47] and to Bart van Merriënboer for a discussion about MIPS (maximum inner product search), which was helpful in extending the proposed approach to word generation.

CHAPTER 8

ON USING VERY LARGE TARGET VOCABULARY FOR NEURAL MACHINE TRANSLATION

8.1 Abstract

Neural machine translation, a recently proposed approach to machine translation based purely on neural networks, has shown promising results compared to the existing approaches such as phrase-based statistical machine translation. Despite its recent success, neural machine translation has its limitation in handling a larger vocabulary, as training complexity as well as decoding complexity increase proportionally to the number of target words. In this paper, we propose a method based on importance sampling that allows us to use a very large target vocabulary without increasing training complexity. We show that decoding can be efficiently done even with the model having a very large target vocabulary by selecting only a small subset of the whole target vocabulary. The models trained by the proposed approach are empirically found to match, and in some cases outperform, the baseline models with a small vocabulary as well as the LSTM-based neural machine translation models. Furthermore, when we use an ensemble of a few models with very large target vocabularies, we achieve performance comparable to the state of the art (measured by BLEU) on both the English→German and English→French translation tasks of WMT'14.

8.2 Introduction

Neural machine translation (NMT) is a recently introduced approach to solving machine translation [2, 54, 99]. In neural machine translation, one builds a single neural network that reads a source sentence and generates its translation. The whole neural network is jointly trained to maximize the conditional probability of a correct translation given a source sentence, using the bilingual corpus. The NMT models have shown to perform as well as the most widely used conventional translation systems [2, 99].

Neural machine translation has a number of advantages over the existing statistical machine translation system, specifically, the phrase-based system [60]. First, NMT requires a minimal set of domain knowledge. For instance, all of the models proposed in [99], [2] or [54] do not assume any linguistic property in both source and target sentences except that they are sequences of words. Second, the whole system is jointly trained to maximize the translation performance, unlike the existing phrase-based system which consists of many separately trained features whose weights are then tuned jointly. Lastly, the memory footprint of the NMT model is often much smaller than the existing system which relies on maintaining large tables of phrase pairs.

Despite these advantages and promising results, there is a major limitation in NMT compared to the existing phrase-based approach. That is, the number of target words must be limited. This is mainly because the complexity of training and using an NMT model increases as the number of target words increases.

A usual practice is to construct a target vocabulary of the K most frequent words (a so-called shortlist), where K is often in the range of $30k$ [2] to $80k$ [99]. Any word not included in this vocabulary is mapped to a special token representing an *unknown* word [UNK]. This approach works well when there are only a few unknown words in the target sentence, but it has been observed that the translation performance degrades rapidly as the number of unknown words increases [2, 20].

In this paper, we propose an approximate training algorithm based on (biased) importance sampling that allows us to train an NMT model with a much larger target vocabulary. The proposed algorithm effectively keeps the computational complexity during training at the level of using only a small subset of the full vocabulary. Once the model with a very large target vocabulary is trained, one can choose to use either all the target words or only a subset of them.

We compare the proposed algorithm against the baseline shortlist-based approach in the tasks of English→French and English→German translation using the NMT model introduced in [2]. The empirical results demonstrate that we can potentially achieve better translation performance using larger vocabularies, and that our approach does not sacrifice too much speed for both training and decoding. Furthermore, we show that the

model trained with this algorithm gets the best translation performance yet achieved by single NMT models on the WMT'14 English→French translation task.

8.3 Neural Machine Translation and Limited Vocabulary Problem

In this section, we briefly describe an approach to neural machine translation proposed recently in [2]. Based on this description we explain the issue of limited vocabularies in neural machine translation.

8.3.1 Neural Machine Translation

Neural machine translation is a recently proposed approach to machine translation, which uses a single neural network trained jointly to maximize the translation performance [2, 21, 34, 54, 99].

Neural machine translation is often implemented as the encoder–decoder network. The encoder reads the source sentence $x = (x_1, \ldots, x_T)$ and encodes it into a sequence of hidden states $h = (h_1, \cdots, h_T)$:

$$h_t = f(x_t, h_{t-1}). \tag{8.1}$$

Then, the decoder, another recurrent neural network, generates a corresponding translation $y = (y_1, \cdots, y_{T'})$ based on the encoded sequence of hidden states h:

$$p(y_t \mid y_{<t}, x) \propto \exp\{q(y_{t-1}, z_t, c_t)\}, \tag{8.2}$$

where

$$z_t = g(y_{t-1}, z_{t-1}, c_t), \tag{8.3}$$

$$c_t = r(z_{t-1}, h_1, \ldots, h_T), \tag{8.4}$$

and $y_{<t} = (y_1, \ldots, y_{t-1})$.

The whole model is jointly trained to maximize the conditional log-probability of

the correct translation given a source sentence with respect to the parameters θ of the model:

$$\theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p(y_t^n \mid y_{<t}^n, x^n),$$

where (x^n, y^n) is the n-th training pair of sentences, and T_n is the length of the n-th target sentence (y^n).

8.3.1.1 Detailed Description

In this paper, we use a specific implementation of neural machine translation that uses an attention mechanism, as recently proposed in [2].

In [2], the encoder in Eq. (8.1) is implemented by a bi-directional recurrent neural network such that

$$h_t = \left[\overleftarrow{h}_t ; \overrightarrow{h}_t \right],$$

where

$$\overleftarrow{h}_t = f\left(x_t, \overleftarrow{h}_{t+1}\right), \overrightarrow{h}_t = f\left(x_t, \overrightarrow{h}_{t-1}\right).$$

They used a gated recurrent unit for f (see, e.g., [21]).

The decoder, at each time, computes the context vector c_t as a convex sum of the hidden states $(h_1, \ldots, h_T)$ with the coefficients $\alpha_1, \ldots, \alpha_T$ computed by

$$\alpha_t = \frac{\exp\{a(h_t, z_{t-1})\}}{\sum_k \exp\{a(h_k, z_{t-1})\}}, \tag{8.5}$$

where a is a feedforward neural network with a single hidden layer.

A new hidden state z_t of the decoder in Eq. (8.3) is computed based on the previous hidden state z_{t-1}, previous generated symbol y_{t-1} and the computed context vector c_t. The decoder also uses the gated recurrent unit, as the encoder does.

The probability of the next target word in Eq. (8.2) is then computed by

$$p(y_t \mid y_{<t}, x) = \frac{1}{Z} \exp\left\{ \mathbf{w}_t^\top \phi\left(y_{t-1}, z_t, c_t\right) + b_t \right\}, \tag{8.6}$$

where ϕ is an affine transformation followed by a nonlinear activation, and $\mathbf{w}_t$ and b_t are respectively the *target word vector* and the target word bias. Z is the normalization constant computed by

$$Z = \sum_{k: y_k \in V} \exp\left\{ \mathbf{w}_k^\top \phi\left(y_{t-1}, z_t, c_t\right) + b_k \right\}, \tag{8.7}$$

where V is the set of all the target words.

For the detailed description of the implementation, we refer the reader to the appendix of [2].

8.3.2 Limited Vocabulary Issue and Conventional Solutions

One of the main difficulties in training this neural machine translation model is the computational complexity involved in computing the target word probability (Eq. (8.6)). More specifically, we need to compute the dot product between the feature $\phi\left(y_{t-1}, z_t, c_t\right)$ and the word vector w_t as many times as there are words in a target vocabulary in order to compute the normalization constant (the denominator in Eq. (8.6)). This has to be done for, on average, 20–30 words per sentence, which easily becomes prohibitively expensive even with a moderate number of possible target words. Furthermore, the memory requirement grows linearly with respect to the number of target words. This has been a major hurdle for neural machine translation, compared to the existing non-parametric approaches such as phrase-based translation systems.

Recently proposed neural machine translation models, hence, use a shortlist of 30k to 80k most frequent words [2, 99]. This makes training more feasible, but comes with a number of problems. First of all, the performance of the model degrades heavily if the translation of a source sentence requires many words that are not included in the shortlist [20]. This also affects the performance evaluation of the system which is often measured

by BLEU. Second, the first issue becomes more problematic with languages that have a rich set of words such as German or other highly inflected languages.

There are two *model-specific* approaches to this issue of large target vocabulary. The first approach is to stochastically approximate the target word probability. This has been proposed recently in [75, 81] based on noise-contrastive estimation [43]. In the second approach, the target words are clustered into multiple classes, or hierarchical classes, and the target probability $p(y_t|y_{<t},x)$ is factorized as a product of the class probability $p(c_t|y_{<t},x)$ and the intra-class word probability $p(y_t|c_t,y_{<t},x)$. This reduces the number of required dot-products into the sum of the number of classes and the words in a class. These approaches mainly aim at reducing the computational complexity during training, but do not often result in speed-up when decoding a translation during test time. [1]

Other than these model-specific approaches, there exist *translation-specific* approaches. A translation-specific approach exploits the properties of the rare target words. For instance, Luong et al. proposed such an approach for neural machine translation [71]. They replace rare words (the words that are not included in the shortlist) in both source and target sentences into corresponding $\langle \text{OOV}_n \rangle$ tokens using the word alignment model. Once a source sentence is translated, each $\langle \text{OOV}_n \rangle$ in the translation will be replaced based on the source word marked by the corresponding $\langle \text{OOV}_n \rangle$.

It is important to note that the model-specific approaches and the translation-specific approaches are often complementary and can be used together to further improve the translation performance and reduce the computational complexity.

8.4 Approximate Learning Approach to Very Large Target Vocabulary

8.4.1 Description

In this paper, we propose a *model-specific* approach that allows us to train a neural machine translation model with a very large target vocabulary. With the proposed approach, the computational complexity of training becomes constant with respect to

1. This is due to the fact that the beam search requires the conditional probability of *every* target word at each time step regardless of the parametrization of the output probability.

the size of the target vocabulary. Furthermore, the proposed approach allows us to efficiently use a fast computing device with limited memory, such as a GPU, to train a neural machine translation model with a much larger target vocabulary.

As mentioned earlier, the computational inefficiency of training a neural machine translation model arises from the normalization constant in Eq. (8.6). In order to avoid the growing complexity of computing the normalization constant, we propose here to use only a small subset V' of the target vocabulary at each update. The proposed approach is based on the earlier work of [7].

Let us consider the gradient of the log-probability of the output in Eq. (8.6). The gradient is composed of a positive and negative part:

$$\begin{aligned} &\nabla \log p(y_t \mid y_{<t}, x) \\ &\quad = \nabla \mathscr{E}(y_t) - \sum_{k:y_k \in V} p(y_k \mid y_{<t}, x) \nabla \mathscr{E}(y_k), \end{aligned} \tag{8.8}$$

where we define the energy $\mathscr{E}$ as

$$\mathscr{E}(y_j) = \mathbf{w}_j^\top \boldsymbol{\phi}\left(y_{j-1}, z_j, c_j\right) + b_j.$$

The second, or negative, term of the gradient is in essence the expected gradient of the energy:

$$\mathbb{E}_P\left[\nabla \mathscr{E}(y)\right], \tag{8.9}$$

where P denotes $p(y \mid y_{<t}, x)$.

The main idea of the proposed approach is to approximate this expectation, or the negative term of the gradient, by importance sampling with a small number of samples. Given a predefined proposal distribution Q and a set V' of samples from Q, we approximate the expectation in Eq. (8.9) with

$$\mathbb{E}_P\left[\nabla \mathscr{E}(y)\right] \approx \sum_{k:y_k \in V'} \frac{\omega_k}{\sum_{k':y_{k'} \in V'} \omega_{k'}} \nabla \mathscr{E}(y_k), \tag{8.10}$$

where

$$\omega_k = \exp\{\mathscr{E}(y_k) - \log Q(y_k)\}. \tag{8.11}$$

This approach allows us to compute the normalization constant during training using only a small subset of the target vocabulary, resulting in much lower computational complexity for each parameter update. Intuitively, at each parameter update, we update only the vectors associated with the correct word $\mathbf{w}_t$ and with the sampled words in V'. Once training is over, we can use the full target vocabulary to compute the output probability of each target word.

Although the proposed approach naturally addresses the computational complexity, using this approach naively does not guarantee that the number of parameters being updated for each sentence pair, which includes multiple target words, is bounded nor can be controlled. This becomes problematic when training is done, for instance, on a GPU with limited memory.

In practice, hence, we partition the training corpus and define a subset V' of the target vocabulary for each partition prior to training. Before training begins, we sequentially examine each target sentence in the training corpus and accumulate unique target words until the number of unique target words reaches the predefined threshold τ. The accumulated vocabulary will be used for this partition of the corpus during training. We repeat this until the end of the training set is reached. Let us refer to the subset of target words used for the i-th partition by V'_i.

This may be understood as having a separate proposal distribution Q_i for each partition of the training corpus. The distribution Q_i assigns equal probability mass to all the target words included in the subset V'_i, and zero probability mass to all the other words, i.e.,

$$Q_i(y_k) = \begin{cases} \frac{1}{|V'_i|} & \text{if } y_t \in V'_i \\ 0 & \text{otherwise.} \end{cases}$$

This choice of proposal distribution cancels out the correction term $-\log Q(y_k)$ from the importance weight in Eqs. (8.10)–(8.11), which makes the proposed approach equivalent to approximating the exact output probability in Eq. (8.6) with

$$p(y_t \mid y_{<t}, x) = \frac{\exp\left\{\mathbf{w}_t^\top \phi\left(y_{t-1}, z_t, c_t\right) + b_t\right\}}{\sum_{k:y_k \in V'} \exp\left\{\mathbf{w}_k^\top \phi\left(y_{t-1}, z_t, c_t\right) + b_k\right\}}.$$

It should be noted that this choice of Q makes the estimator biased.

The proposed procedure results in speed up against usual importance sampling, as it exploits the advantage of modern computers in doing matrix-matrix vs matrix-vector multiplications.

8.4.1.1 Informal Discussion on Consequence

The parametrization of the output probability in Eq. (8.6) can be understood as arranging the vectors associated with the target words such that the dot product between the most likely, or correct, target word's vector and the current hidden state is maximized. The exponentiation followed by normalization is simply a process in which the dot products are converted into proper probabilities.

As learning continues, therefore, the vectors of all the likely target words tend to align with each other but not with the others. This is achieved exactly by moving the vector of the correct word in the direction of $\phi\left(y_{t-1}, z_t, c_t\right)$, while pushing all the other vectors away, which happens when the gradient of the logarithm of the exact output probability in Eq. (8.6) is maximized. Our approximate approach, instead, moves the word vectors of the correct words and of only a subset of sampled target words (those included in V').

8.4.2 Decoding

Once the model is trained using the proposed approximation, we can use the full target vocabulary when decoding a translation given a new source sentence. Although

this is advantageous as it allows the trained model to utilize the whole vocabulary when generating a translation, doing so may be too computationally expensive, e.g., for real-time applications.

Since training puts the target word vectors in the space so that they align well with the hidden state of the decoder only when they are likely to be a correct word, we can use only a subset of candidate target words during decoding. This is similar to what we do during training, except that at test time, we do not have access to a set of correct target words.

The most naïve way to select a subset of candidate target words is to take only the top-K most frequent target words, where K can be adjusted to meet the computational requirement. This, however, effectively cancels out the whole purpose of training a model with a very large target vocabulary. Instead, we can use an existing word alignment model to align the source and target words in the training corpus and build a dictionary. With the dictionary, for each source sentence, we construct a target word set consisting of the K-most frequent words (according to the estimated unigram probability) and, using the dictionary, at most K' likely target words for each source word. K and K' may be chosen either to meet the computational requirement or to maximize the translation performance on the development set. We call a subset constructed in either of these ways a *candidate list*.

8.4.3 Source Words for Unknown Words

In the experiments, we evaluate the proposed approach with the neural machine translation model called RNNsearch [2] (see Sec. 8.3.1.1). In this model, as a part of decoding process, we obtain the alignments between the target words and source locations via the alignment model in Eq. (8.5).

We can use this feature to infer the source word to which each target word was most aligned (indicated by the largest α_t in Eq. (8.5)). This is especially useful when the model generated an [UNK] token. Once a translation is generated given a source sentence, each [UNK] may be replaced using a translation-specific technique based on the aligned source word. For instance, in the experiment, we try replacing each [UNK]

token with the aligned source word or its most likely translation determined by another word alignment model. Other techniques such as transliteration may also be used to further improve the performance [59].

8.5 Experiments

We evaluate the proposed approach in English→French and English→German translation tasks. We trained the neural machine translation models using only the bilingual, parallel corpora made available as a part of WMT'14. For each pair, the datasets we used are:

– English→French: [2]

  Europarl v7, Common Crawl, UN, News Commentary, Gigaword

– English→German:

  Europarl v7, Common Crawl, News Commentary

To ensure fair comparison, the English→French corpus, which comprises approximately 12 million sentences, is identical to the one used in [2, 54, 99]. As for English→German, the corpus was preprocessed, in a manner similar to [70, 90], in order to remove many poorly translated sentences.

We evaluate the models on the WMT'14 test set (news-test 2014), [3] while the concatenation of news-test-2012 and news-test-2013 is used for model selection (development set). Table 8.I presents data coverage w.r.t. the vocabulary size, on the target side.

Unless mentioned otherwise, all reported BLEU scores [88] are computed with the multi-bleu.perl script [4] on the cased tokenized translations.

2. The preprocessed data can be found and downloaded from `http://www-lium.univ-lemans.fr/~schwenk/nnmt-shared-task/README`.

3. To compare with previous submissions, we use the filtered test sets.

4. `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

	English-French		English-German	
	Train	Test	Train	Test
15k	93.5	90.8	88.5	83.8
30k	96.0	94.6	91.8	87.9
50k	97.3	96.3	93.7	90.4
500k	99.5	99.3	98.4	96.1
All	100.0	99.6	100.0	97.3

Table 8.I: Data coverage (in %) on target-side corpora for different vocabulary sizes. "All" refers to all the tokens in the training set.

8.5.1 Settings

As a baseline for English→French translation, we use the **RNNsearch** model proposed by [2], with 30k source and target words. [5] Another RNNsearch model is trained for English→German translation with 50k source and target words.

For each language pair, we train another set of RNNsearch models with much larger vocabularies of 500k source and target words, using the proposed approach. We call these models **RNNsearch-LV**. We vary the size of the shortlist used during training (τ in Sec. 8.4.1). We tried 15k and 30k for English→French, and 15k and 50k for English→German. We later report the results for the best performance on the development set, with models generally evaluated every twelve hours. The training speed is approximately the same as for RNNsearch. Using a 780 Ti or Titan Black GPU, we could process 100k mini-batches of 80 sentences in about 29 and 39 hours respectively for $\tau = 15k$ and $\tau = 50k$.

For both language pairs, we also trained new models, with $\tau = 15k$ and $\tau = 50k$, by reshuffling the dataset at the beginning of each epoch. While this causes a non-negligible amount of overhead, such a change allows words to be contrasted with different sets of other words each epoch.

To stabilize parameters other than the word embeddings, at the end of the training stage, we freeze the word embeddings and tune only the other parameters for approximately two more days after the peak performance on the development set is observed.

5. The authors of [2] gave us access to their trained models. We chose the best one on the validation set and resumed training.

This helped increase BLEU scores on the development set.

We use beam search to generate a translation given a source. During beam search, we keep a set of 12 hypotheses and normalize probabilities by the length of the candidate sentences, as in [20]. [6] The candidate list is chosen to maximize the performance on the development set, for $K \in \{15k, 30k, 50k\}$ and $K' \in \{10, 20\}$. As explained in Sec. 8.4.2, we test using a bilingual dictionary to accelerate decoding and to replace unknown words in translations. The bilingual dictionary is built using *fast_align* [30]. We use the dictionary only if a word starts with a lowercase letter, and otherwise, we copy the source word directly. This led to better performance on the development sets.

Note on ensembles For each language pair, we began training four models from each of which two points corresponding to the best and second-best performance on the development set were collected. We continued training from each point, while keeping the word embeddings fixed, until the best development performance was reached, and took the model at this point as a single model in an ensemble. This procedure resulted in a total of eight models from which we averaged the length-normalized log-probabilities. Since much of training had been shared, the composition of such ensembles may be sub-optimal. This is supported by the fact that higher cross-model BLEU scores [36] are observed for models that were partially trained together.

8.5.2 Translation Performance

In Table 8.II, we present the results obtained by the trained models with very large target vocabularies, and alongside them, the previous results reported in [99], [71], [15] and [28]. Without translation-specific strategies, we can clearly see that the RNNsearch-LV outperforms the baseline RNNsearch.

In the case of the English→French task, RNNsearch-LV approached the performance level of the previous best single neural machine translation (NMT) model, even without any translation-specific techniques (Sec. 8.4.2–8.4.3). With these, however, the RNNsearch-LV outperformed it. The performance of the RNNsearch-LV is also better

6. These experimental details differ from [2].

	RNNsearch	RNNsearch-LV	Google	Phrase-based SMT	
Basic NMT	29.97 (26.58)	32.68 (28.76)	30.6*	33.3*	37.03$^\bullet$
+Candidate List	–	33.36 (29.32)	–		
+UNK Replace	33.08 (29.08)	34.11 (29.98)	33.1°		
+Reshuffle (τ=50k)	–	34.60 (30.53)	–		
+Ensemble	–	37.19 (31.98)	37.5°		

(a) English→French

	RNNsearch	RNNsearch-LV	Phrase-based SMT
Basic NMT	16.46 (17.13)	16.95 (17.85)	20.67$^\diamond$
+Candidate List	–	17.46 (18.00)	
+UNK Replace	18.97 (19.16)	18.89 (19.03)	
+Reshuffle	–	19.40 (19.37)	
+Ensemble	–	21.59 (21.06)	

(b) English→German

Table 8.II: The translation performances in BLEU obtained by different models on (a) English→French and (b) English→German translation tasks. RNNsearch is the model proposed in [2], RNNsearch-LV is the RNNsearch trained with the approach proposed in this paper, and Google is the LSTM-based model proposed in [99]. Unless mentioned otherwise, we report single-model RNNsearch-LV scores using $\tau = 30k$ (English→French) and $\tau = 50k$ (English→German). For the experiments we have run ourselves, we show the scores on the development set as well in the brackets. ($\star$) [99], ($\circ$) [71], ($\bullet$) [28], ($*$) Standard Moses Setting [21], ($\diamond$) [15].

	CPU*	GPU$^\circ$
RNNsearch	0.09 s	0.02 s
RNNsearch-LV	0.80 s	0.25 s
RNNsearch-LV +Candidate list	0.12 s	0.05 s

Table 8.III: The average per-word decoding time. Decoding here does not include parameter loading and unknown word replacement. The baseline uses 30k words. The candidate list is built with $K = 30k$ and $K' = 10$. ($\star$) i7-4820K (single thread), ($\circ$) GTX TITAN Black

than that of a standard phrase-based translation system [21]. Furthermore, by combining 8 models, we were able to achieve a translation performance comparable to the state of the art, measured in BLEU.

For English→German, the RNNsearch-LV outperformed the baseline before unknown word replacement, but after doing so, the two systems performed similarly. We could reach higher large-vocabulary single-model performance by reshuffling the dataset, but this step could potentially also help the baseline. In this case, we were able to surpass the previously reported best translation result on this task by building an ensemble of 8 models.

With $\tau = 15k$, the RNNsearch-LV performance worsened a little, with best BLEU scores, without reshuffling, of 33.76 and 18.59 respectively for English→French and English→German.

The English→German ensemble described in this paper has also been used for the shared translation task of the 10^{th} Workshop on Statistical Machine Translation (WMT'15), where it was ranked first in terms of BLEU score. The translations by this ensemble can be found online. [7]

8.5.3 Analysis

8.5.3.1 Decoding Speed

In Table 8.III, we present the timing information of decoding for different models. Clearly, decoding from RNNsearch-LV with the full target vocabulary is slowest. If we use a candidate list for decoding each translation, the speed of decoding substantially improves and becomes close to the baseline RNNsearch.

A potential issue with using a candidate list is that for each source sentence, we must re-build a target vocabulary and subsequently replace a part of the parameters, which may easily become time-consuming. We can address this issue, for instance, by building a common candidate list for multiple source sentences. By doing so, we were able to match the decoding speed of the baseline RNNsearch model.

7. `http://matrix.statmt.org/matrix/output/1774?run_id=4079`

8.5.3.2 Decoding Target Vocabulary

For English→French ($\tau = 30k$), we evaluate the influence of the target vocabulary when translating the test sentences by using the union of a fixed set of $30k$ common words and (at most) K' likely candidates for each source word according to the dictionary. Results are presented in Figure 8.1. With $K' = 0$ (not shown), the performance of the system is comparable to the baseline when not replacing the unknown words (30.12), but there is not as much improvement when doing so (31.14). As the large vocabulary model does not predict [UNK] as much during training, it is less likely to generate it when decoding, limiting the effectiveness of the post-processing step in this case. With $K' = 1$, which limits the diversity of allowed uncommon words, BLEU is not as good as with moderately larger K', which indicates that our models can, to some degree, correctly choose between rare alternatives. If we rather use $K = 50k$, as we did for testing based on validation performance, the improvement over $K' = 1$ is approximately 0.2 BLEU.

When validating the choice of K, we found it to be correlated with the value of τ used during training. For example, on the English→French validation set, with $\tau = 15k$ (and $K' = 10$), the BLEU score is 29.44 with $K = 15k$, but drops to 29.19 and 28.84 respectively for $K = 30k$ and $50k$. For $\tau = 30k$, the score increases moderately from $K = 15k$ to $K = 50k$. A similar effect was observed for English→German and on the test sets. As our implementation of importance sampling does not apply the usual correction to the gradient, it seems beneficial for the test vocabularies to resemble those used during training.

8.6 Conclusion

In this paper, we proposed a way to extend the size of the target vocabulary for neural machine translation. The proposed approach allows us to train a model with much larger target vocabulary without any substantial increase in computational complexity. It is based on the earlier work in [7] which used importance sampling to reduce the complexity of computing the normalization constant of the output word probability in

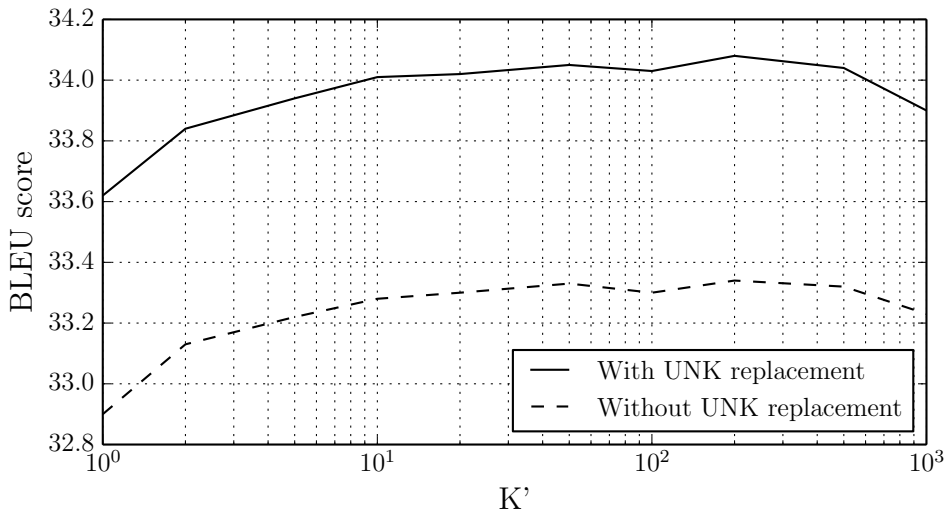

Figure 8.1: Single-model test BLEU scores (English→French) with respect to the number of dictionary entries K' allowed for each source word.

neural language models.

On English→French and English→German translation tasks, we observed that the neural machine translation models trained using the proposed method performed as well as, or better than, those using only limited sets of target words, even when replacing unknown words. As performance of the RNNsearch-LV models increased when only a selected subset of the target vocabulary was used during decoding, this makes the proposed learning algorithm more practical.

When measured by BLEU, our models showed translation performance comparable to the state-of-the-art translation systems on both the English→French task and English→German task. On the English→French task, a model trained with the proposed approach outperformed the best single neural machine translation (NMT) model from [71] by approximately 1 BLEU point. The performance of the ensemble of multiple models, despite its relatively less diverse composition, is approximately 0.3 BLEU points away from the best system [71]. On the English→German task, the best performance of 21.59 BLEU by our model is higher than that of the previous state of the art (20.67) reported in [15].

Finally, we release the source code used in our experiments to encourage progress in neural machine translation. [8]

Acknowledgments

The authors would like to thank the developers of Theano [5, 12]. We acknowledge the support of the following agencies for research funding and computing support: NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs, CIFAR and Samsung.

8. https://github.com/sebastien-j/LV_groundhog

CHAPTER 9

CONCLUSION

In this thesis, we have conducted various experiments on word embeddings and neural machine translation. On the task of word analogies, we have confirmed previous findings indicating the usefulness of position-dependent weights, highlighted that current analogy recovery methods are suboptimal and presented preliminary work on learning from analogies directly. Motivated by properties of NMT word representations, we have then shown that very simple bilingual embeddings could reach state-of-the-art accuracy on two word similarity tasks. Finally, we have proposed a GPU-friendly approach based on importance sampling to extend neural machine translation to large vocabularies. The proposed techniques use little GPU memory and are reasonably fast for both training and word generation.

BIBLIOGRAPHY

[1] Jacob Andreas and Dan Klein. When and why are log-linear models self-normalizing? In *Proc. NAACL'2015*, 2015.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. Technical report, arXiv preprint arXiv:1409.0473, 2014.

[3] Rajendra Banjade, Nabin Maharjan, NobalB. Niraula, Vasile Rus, and Dipesh Gautam. Lemon and tea are not similar: Measuring word-to-word similarity by combining different methods. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 9041 of *Lecture Notes in Computer Science*, pages 335–346. Springer International Publishing, 2015. ISBN 978-3-319-18110-3. doi: 10.1007/978-3-319-18111-0_25. URL http://dx.doi.org/10.1007/978-3-319-18111-0_25.

[4] Colin Bannard and Chris Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604. Association for Computational Linguistics, 2005.

[5] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[6] Yoshua Bengio and Jean-Sébastien Sénécal. Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the conference on Artificial Intelligence and Statistics (AISTATS)*, 2003.

[7] Yoshua Bengio and Jean-Sébastien Sénécal. Adaptive importance sampling to

accelerate training of a neural probabilistic language model. *IEEE Trans. Neural Networks*, 19(4):713–722, 2008.

[8] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[9] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[10] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[11] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. Technical report, arXiv:1206.5538, 2012.

[12] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[13] P. F. Brown, V. J. Della Pietra, P. V. DeSouza, J. C. Lai, and R. L. Mercer. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

[14] Elia Bruni, Nam-Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *Journal of Artificial Intelligence Research (JAIR)*, 49:1–47, 2014.

[15] Christian Buck, Kenneth Heafield, and Bas van Ooyen. N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, Reykjavík, Iceland, May 2014.

[16] John A Bullinaria and Joseph P Levy. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and svd. *Behavior research methods*, 44(3):890–907, 2012.

[17] John Caron. Experiments with lsa scoring: optimal rank and basis. In *Computational information retrieval*, pages 157–169. Society for Industrial and Applied Mathematics, 2001.

[18] Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861, 2014.

[19] Stanley F. Chen and Joshua T. Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.

[20] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–Decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, October 2014.

[21] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*, October 2014.

[22] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[23] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[24] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

[25] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[26] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In *Proc. ACL'2014*, 2014.

[27] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 2011.

[28] Nadir Durrani, Barry Haddow, Philipp Koehn, and Kenneth Heafield. Edinburgh's phrase-based machine translation systems for WMT-14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 97–104. Association for Computational Linguistics Baltimore, MD, USA, 2014.

[29] Chris Dyer. Notes on noise contrastive estimation and negative sampling. *arXiv preprint arXiv:1410.8251*, 2014.

[30] Chris Dyer, Victor Chahuneau, and Noah A. Smith. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[31] Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*, volume 2014, 2014.

[32] Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*, 2015.

[33] J. Firth. *A synopsis of linguistic theory 1930-1955*. Studies in Linguistic Analysis, Philological. Longman, 1957.

[34] Mikel L. Forcada and Ramón P. Ñeco. Recursive hetero-associative memories for translation. In José Mira, Roberto Moreno-Díaz, and Joan Cabestany, editors, *Biological and Artificial Computation: From Neuroscience to Technology*, volume 1240 of *Lecture Notes in Computer Science*, pages 453–462. Springer Berlin Heidelberg, 1997.

[35] Dayne Freitag, Matthias Blume, John Byrnes, Edmond Chow, Sadik Kapadia, Richard Rohwer, and Zhiqiang Wang. New experiments in distributional representations of synonymy. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 25–32. Association for Computational Linguistics, 2005.

[36] Markus Freitag, Stephan Peitz, Joern Wuebker, Hermann Ney, Matthias Huck, Rico Sennrich, Nadir Durrani, Maria Nadejde, Philip Williams, Philipp Koehn, et al. Eu-bridge MT: Combined machine translation. *ACL 2014*, page 105, 2014.

[37] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL `http://cs.jhu.edu/~ccb/publications/ppdb.pdf`.

[38] Bin Gao, Jiang Bian, and Tie-Yan Liu. Wordrep: A benchmark for research on learning word representations. *arXiv preprint arXiv:1407.1640*, 2014.

[39] Eva Martínez Garcia, Cristina España-Bonet, Jörg Tiedemann, and Lluíz Màrquez. Word's vector representations meet machine translation. In *Eighth*

Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), pages 132–134, 2014.

[40] J. Goodman. Classes for fast maximum entropy training. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Utah, 2001.

[41] Stephan Gouws, Yoshua Bengio, and Greg Corrado. Bilbowa: Fast bilingual distributed representations without word alignments. Technical report, arXiv:1410.2455, 2014.

[42] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18 (5):602–610, 2005.

[43] M. Gutmann and A. Hyvarinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS'10)*, 2010.

[44] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[45] Karl Moritz Hermann and Phil Blunsom. Multilingual Distributed Representations without Word Alignment. In *Proceedings of ICLR*, apr 2014.

[46] Karl Moritz Hermann and Phil Blunsom. Multilingual Models for Compositional Distributional Semantics. In *Proceedings of ACL*, June 2014. URL `http://arxiv.org/abs/1404.4641`.

[47] Felix Hill, KyungHyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. Not all neural embeddings are born equal. Technical report, arXiv:1410.0718, 2014.

[48] Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. Embedding word similarity with neural machine translation. *arXiv preprint arXiv:1412.6448*, 2014.

[49] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *arXiv preprint arXiv:1408.3456*, 2014.

[50] Geoffrey E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12, Amherst 1986, 1986. Lawrence Erlbaum, Hillsdale.

[51] Geoffrey E Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network.

[52] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[53] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014.

[54] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709. Association for Computational Linguistics, 2013.

[55] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[56] Ryan Kiros, Richard Zemel, and Ruslan R Salakhutdinov. A multiplicative model for learning distributed text-based attribute representations. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2014.

[57] A. Klementiev, I. Titov, and B. Bhattarai. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, 2012.

[58] Reinhard Kneser and Hermann Ney. Improved backing-off for M-gram language modeling. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 181–184, 1995.

[59] Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[60] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, 2003.

[61] Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. Learning Bilingual Word Representations by Marginalizing Alignments. In *Proceedings of ACL*, June 2014. URL `http://arxiv.org/abs/1405.0947`.

[62] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS'2012)*. 2012.

[63] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. In *NIPS'94*, pages 231–238. Cambridge MA: MIT Press, 1995.

[64] Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.

[65] Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[66] Omer Levy and Yoav Goldberg. Dependencybased word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308, 2014.

[67] Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. *CoNLL-2014*, page 171, 2014.

[68] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

[69] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association of Computational Linguistics (TACL)*, 2015.

[70] Liangyou Li, Xiaofeng Wu, Santiago Cortes Vaillo, Jun Xie, Andy Way, and Qun Liu. The DCU-ICTCAS MT system at WMT 2014 on German-English translation task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 136–141, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W14/W14-3314`.

[71] Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*, 2014.

[72] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

[73] Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Cernocky. Empirical evaluation and combination of advanced language modeling techniques. In *Proc. 12th annual conference of the international speech communication association (INTERSPEECH 2011)*, 2011.

[74] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Proc. 2011 IEEE international conference on acoustics, speech and signal processing (ICASSP 2011)*, 2011.

[75] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations: Workshops Track*, 2013.

[76] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. Technical report, arXiv:1309.4168, 2013.

[77] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[78] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[79] Andriy Mnih and Geoffrey E. Hinton. Three new graphical models for statistical language modelling. In Zoubin Ghahramani, editor, *ICML 2007*, pages 641–648. ACM, 2007.

[80] Andriy Mnih and Geoffrey E. Hinton. A scalable hierarchical distributed language model. In *NIPS'08*, pages 1081–1088, 2009.

[81] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5165-learning-word-embeddings-efficiently-with-noise-contrastive.pdf`.

[82] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273, 2013.

[83] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

[84] Frédéric Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In Robert G. Cowell and Zoubin Ghahramani, editors, *AISTATS'2005*, pages 246–252, 2005.

[85] Yoshiki Niwa and Yoshihiko Nitta. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 304–309. Association for Computational Linguistics, 1994.

[86] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51, 2003.

[87] Sebastian Padó and Mirella Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, 2007.

[88] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[89] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML'13)*. ACM, 2013. URL `http://icml.cc/2013/`.

[90] Stephan Peitz, Joern Wuebker, Markus Freitag, and Hermann Ney. The RWTH Aachen German-English machine translation system for WMT 2014. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 157–162, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3317.

[91] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empiricial Methods in Natural Language Processing*, 2014.

[92] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[93] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[94] Holger Schwenk. Efficient training of large neural networks for language modeling. In *International Joint Conference on Neural Networks (IJCNN)*, volume 4, pages 3050–3064, 2004.

[95] Nakatani Shuyo. Language detection library for java, 2010. URL http://code.google.com/p/language-detection/.

[96] Richard Socher, Christopher Manning, and Andrew Y. Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning (ICML'2011)*, 2011.

[97] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[98] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[99] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS'2014*, 2014.

[100] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics(ACL2010)*, pages 384–394. Association for Computational Linguistics, July 2010.

[101] Peter D Turney. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, pages 533–585, 2012.

[102] Peter D Turney, Patrick Pantel, et al. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188, 2010.

[103] Laurens van der Maaten. Barnes-hut-sne. *arXiv preprint arXiv:1301.3342*, 2013.

[104] Lonneke Van der Plas and Jörg Tiedemann. Finding synonyms using automatic word alignment and measures of distributional similarity. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 866–873. Association for Computational Linguistics, 2006.

[105] Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. Decoding with large-scale neural language models improves translation. Citeseer.

[106] Lior Wolf, Yair Hanani, Kfir Bar, and Nachum Dershowitz. Joint word2vec networks for bilingual semantic representations. *International Journal of Computational Linguistics and Applications*, 5(1):27–44, 2014.

[107] Hua Wu and Ming Zhou. Optimizing synonym extraction using monolingual and bilingual resources. In *Proceedings of the second international workshop on*

Paraphrasing-Volume 16, pages 72–79. Association for Computational Linguistics, 2003.

[108] Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1219–1228. ACM, 2014.

[109] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. Technical report, arXiv 1212.5701, 2012. URL `http://arxiv.org/abs/1212.5701`.

[110] Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, 2013. URL `http://ai.stanford.edu/~wzou/emnlp2013_ZouSocherCerManning.pdf`.

Appendix I

Analogy questions: Examples

Google	a	b	c	d
capital-common-countries	athens	greece	baghdad	iraq
capital-world	abuja	nigeria	accra	ghana
currency	algeria	dinar	angola	kwanza
city-in-state	chicago	illinois	houston	texas
family	boy	girl	brother	sister
gram1-adjective-to-adverb	amazing	amazingly	apparent	apparently
gram2-opposite	acceptable	unacceptable	aware	unaware
gram3-comparative	bad	worse	big	bigger
gram4-superlative	bad	worst	big	biggest
gram5-present-participle	code	coding	dance	dancing
gram6-nationality-adjective	albania	albanian	argentina	argentinean
gram7-past-tense	dancing	danced	decreasing	decreased
gram8-plural	banana	bananas	bird	birds
gram9-plural-verbs	decrease	decreases	describe	describes
MSR				
adj. base/comparative	good	better	rough	rougher
adj. comparative/superlative	simplest	simpler	slimmest	slimmer
adj. base/superlative	bright	brightest	big	biggest
nouns singular/plural	year	years	law	laws
verbs base/past	be	was	return	returned
verbs past/third person	brings	brought	believes	believed
verbs base/third person	recognize	recognizes	keep	keeps

Table I.I: Examples of analogy questions

Appendix II

Analogy questions: Average cosine similarity

	a		b		c	
Google	BOW	PDW	BOW	PDW	BOW	PDW
capital-common-countries	0.21	0.31	0.42	0.56	0.65	0.62
capital-world	0.16	0.24	0.33	0.46	0.65	0.60
currency	0.08	0.10	0.24	0.29	0.28	0.25
city-in-state	0.33	0.30	0.50	0.50	0.68	0.56
family	0.39	0.32	0.49	0.41	0.77	0.75
gram1-adjective-to-adverb	0.16	0.12	0.24	0.33	0.62	0.39
gram2-opposite	0.17	0.20	0.29	0.34	0.54	0.51
gram3-comparative	0.18	0.18	0.39	0.42	0.64	0.62
gram4-superlative	0.08	0.15	0.30	0.42	0.53	0.53
gram5-present-participle	0.12	0.13	0.16	0.25	0.75	0.65
gram6-nationality-adjective	0.21	0.13	0.33	0.54	0.70	0.44
gram7-past-tense	0.09	0.14	0.18	0.29	0.69	0.63
gram8-plural	0.13	0.15	0.18	0.28	0.71	0.62
gram9-plural-verbs	0.10	0.11	0.22	0.22	0.66	0.60
total	0.17	0.19	0.31	0.40	0.64	0.56

	a		b		c	
MSR	BOW	PDW	BOW	PDW	BOW	PDW
adj. base/comparative	0.10	0.16	0.27	0.34	0.50	0.51
adj. comparative/superlative	0.10	0.15	0.27	0.35	0.50	0.50
adj. base/superlative	0.09	0.14	0.24	0.32	0.52	0.52
nouns singular/plural	0.06	0.10	0.10	0.19	0.69	0.62
verbs base/past	0.09	0.16	0.29	0.33	0.57	0.62
verbs past/third person	0.07	0.15	0.24	0.32	0.57	0.61
verbs base/third person	0.08	0.16	0.25	0.32	0.58	0.62
total	0.08	0.15	0.24	0.31	0.56	0.57

Table II.I: For word analogy questions of the form $a : b :: c : d$, average cosine similarity of question words a, b, c with the expected answer d, either without (BOW) or with (PDW) position-dependent weights